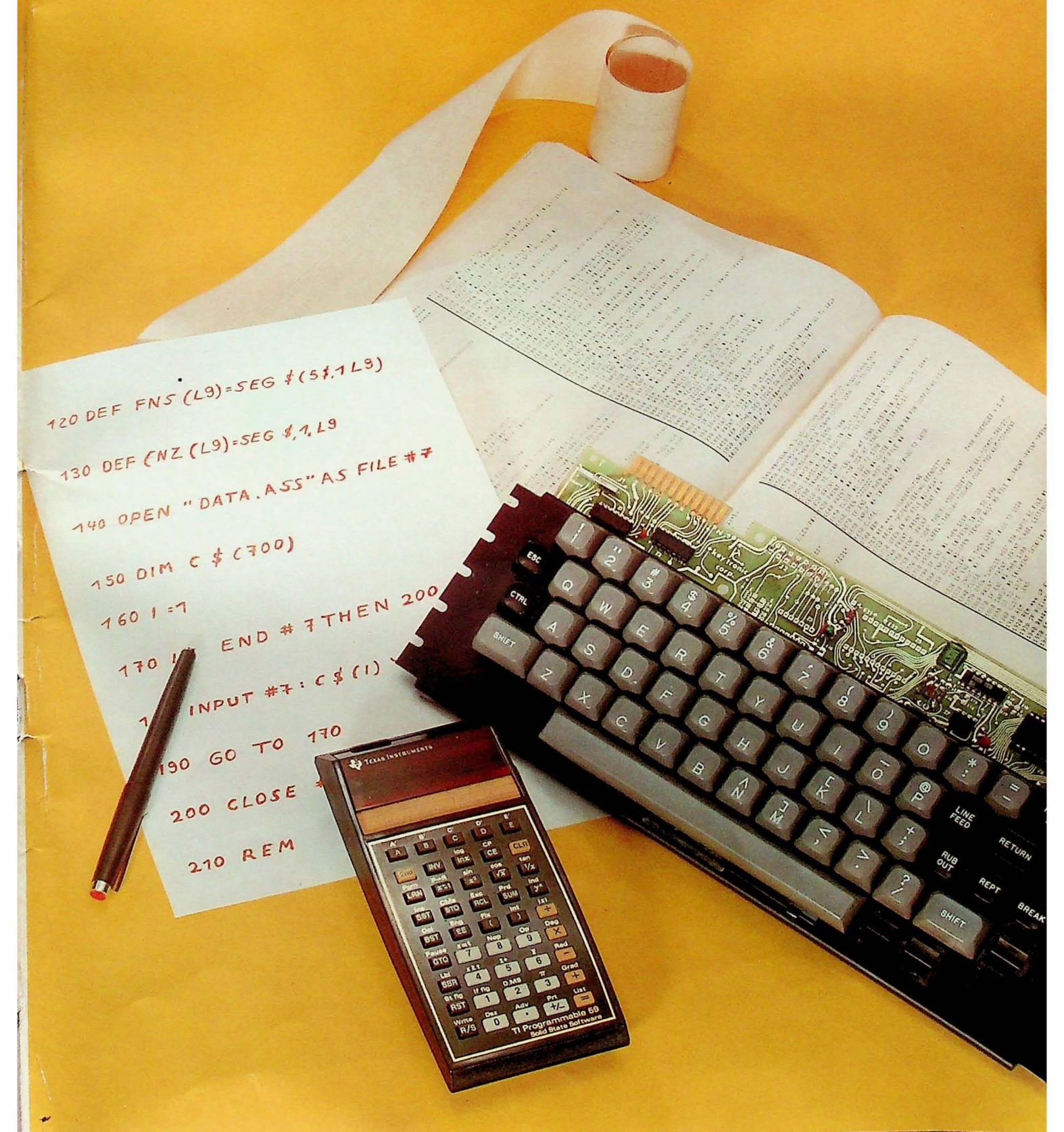
Funkschau
Sonderheft Nr. 31
Preis 14.80 DM

programme

für Kleincomputer und Taschenrechner



Anpassung von Programmen an unterschiedliche Computer

Die folgende Übersicht soll das Umschreiben von Basic- und Maschinenprogrammen für den eigenen Computertyp erleichtern. Sie kann jedoch keinen Anspruch auf Vollständigkeit erheben.

TRS-80-Version (Level 2)	Entsprechende PET-/CBM-Version
10 CLS: REM Schirm löschen	10 PRINT"s": REM Taste Clear
15 INKEY A$	15 GETA$: IFA$="" GOTO15
20 LPRINT "Tast"	20 OPEN 1,4: PRINT#1, "Test": CLOSE 1
30 OPEN "R", 2, "ADDRESS"	30 OPEN 2,1,0, "ADDRESS" (2 = File-Nr., 1 = IEC-Kanal, 0 = Lesen)
620 IFINSTR (N$, C$) = 0THEN600	620 K = LEN (C$: S=LEN (N$)-K 621 IFS < 0 GOTO600 622 FOR I=1TO S+1: IFMID$ (N$, I, K)=C$GOTO630 623 NEXT 624 GOTO600
50 DEFINT A–Z	TRS-80-Variablendefinition; kann bei PET/CBM entfallen
60 PRINT STRING$ (3, CHR$(24));	60 FOR I=1 TO3: PRINTCHR$ (24);: NEXT (Vorsicht: Die Zeichencodes von TRS-80 und PET/CBM sind nicht identisch!)
2010 IF A=B THEN 2050 ELSE NEXT	2010 IF A<>B THEN NEXT
2050 H=0: CU=N	2050 H=0: CU=N

Systemadressen von 6502-Computern

KIM-1	*AIM-65*	*PET/CBM*	*Bedeutung*
1EA0	E9BC	FFD2	ASCII-Zeichen ausgeben
1E5A	E993	FFCF	ASCII-Zeichen eingeben
1700	A001, A00F	E841, E84F	Port A, Datenregister
1701	A003	E843	Port A, Richtungsregister
1702	A000	E840	Port B, Datenregister
1703	A002	E842	Port B, Richtungsregister
–	A004-A00E	E844-E84E	VIA-Timer 6522 [1]
1704–170F	–	–	RRIOT-Timer 6530 [2]
1E2F	E9F0	–	Ausgabe von CRLF (neue Zeile)
1C4F	E182	CBM: 0400	Rückkehr ins Monitorprogramm [3]
17FE, 17FF	A404, A405	0219, 021A	IRQ-Vektor (VIA etc.)
17FA, 17FB	A402, A403	unbenutzt	NMI-Vektor

Weitere Angaben:

[1] Das VIA 6522. Hobbycomputer 2, Sonderheft des Franzis-Verlages.
[2] Anwendungsbeispiele für den Mikroprozessor 6502. RPB 173, Franzis-Verlag.
[3] AIM-65-Monitor-Program-Listing. Fa. Rockwell, 8033 Martinsried.

Vorwort

Programme für so unterschiedliche Geräte wie Taschenrechner, Basic-Computer und „nackte" Mikroprozessoren in einem Heft zusammenzuwürfeln, bedarf sicher einer Erklärung.

Viele „alte Hasen" haben ihre ersten Programmierkenntnisse mit Taschenrechnern erlangt. Schon diese Vorstufe zu Kompaktcomputern zwingt dazu, sich die streng logische und konsequente Denkweise anzueignen, die allen Programmen eigen ist – oder sein sollte. Nach einigen Monaten erkennt man dann die Grenzen von Taschenrechnern; ein Dialogbetrieb ist ebenso wie eine Textverarbeitung nur sehr unvollkommen möglich, und weil komplexere Programme nun einmal den Dialog brauchen, um überhaupt bedienbar zu werden, ist die nächste Anschaffung oft ein Basic-Computer.

Doch auch die Grenzen von Basic sind bald erkannt: eingeschränkte Verarbeitungsgeschwindigkeit auch primitivster Aufgaben, umständliche Programmierung bei Abfrage und Steuerung externer Geräte. Oft stellt der Anwender dann fest, daß es vielleicht günstiger gewesen wäre, sich einen Computer zuzulegen, den man auch in der prozessoreigenen Maschinensprache komfortabel programmieren kann und der sich notfalls mit einem Basic-Interpreter nachrüsten läßt. Denn nur in Maschinensprache lassen sich all jene augeklügelten Eigenschaften und Befehle verwerten, die einen Mikroprozessor modern erscheinen lassen.

Das vorliegende Sonderheft bietet dem Leser nicht nur Programme für alle denkbaren Anwendungsgebiete von Taschenrechnern und Computern, sondern zeigt ihm auch, wie sich ein und dasselbe Problem mit unterschiedlichen Programmiersprachen lösen läßt. Damit hat auch der „Newcomer" eine Chance, Komfort, Verwendungsmöglichkeiten und Grenzen der drei genannten Programmiertechniken objektiv zu vergleichen.

Die Redaktion

Inhalt

Programme für Taschenrechner

Allround-Programm für die Basisumwandlung (HP-25) . 7
TI-59 berechnet Strahlungswiderstand 11
Berechnung der Lohn- und Einkommensteuer mit dem HP-67 13
HP-41 druckt Kalender 16
Kalenderausdruck mit TI-59 17
TI-59 berechnet Spulen 19
TI-59 dimensioniert Verstärker 21
„Notenassistent" sorgt für Zeugnistransparenz (HP-41) 23
TI-59-Mehrfachplotter 26
TI-59 plottet Funktionen 29
Primfaktor-Zerlegung mit dem TI-59 31
HP-41-Tips 52
HP-25 berechnet Frequenzgang 63
Hexadezimal-Dezimal-Umwandlung mit dem TI-59 81

Basic-Programme

Berechnung von Yagi-Antennen (PET) 12
PET, der Unergründete 18
Hex-Monitor für den TRS-80 20
„Notenassistent" sorgt für Zeugnis-Transparenz 23
Flugvorbereitung (PET) 27
Kurvendiskussion in Basic (AIM-65) 30
Linker Rand bei LLIST und LPRINT (TRS-80) 32
Minotaurus per Computer (TRS-80) 33
PET plottet Funktionen 34
AIM-65-Tips 35
Dateiverarbeitung mit dem TRS-80 36
TRS-80 – ein Computer mit kleinem Fehler 37
Datenbank-Programm für den TRS-80 38
Superhirn (PET) 41
Kalender-Algorithmus (PET, TRS-80) 42
„INGET" verkürzt Tipparbeit (PET) 43
Bit-Flags ermöglichen elegante Basic-Wege (PET, TRS-80) 45
TRS-80 macht Musik 46
Entfernungsberechnung mit TRS-80 Level I 48
Stichwortsuche in Basic (PET) 48
Morsedecoder in Basic (MZ-80 K) 59

Maschinensprache-Software

Hex-BCD-Umwandlung mit dem Z-80 10
Textausdruck mit Baudot-Fernschreiber (KIM) 44
TRS-80 macht Musik 46
Plotten mit dem AIM-Drucker 47
6502-Texteditor 49
Erweitertes Betriebssystem für den Nascom-1 53
AIM-Directory 55
KIM als Morsetrainer 56
FSK-Demodulation per Software (6502) 57
Ein Software-Tondecoder (6502) 58
KIM-Monitorprogramm für Baudot-Fernschreiber 60
Software-Uhr für den Z-80-Kit 62
6502 simuliert 8080 64
KIM spielt Lotto 65
KIM als Siebensegment-Fernschreiber 66
Von A bis Z: Alphabetisches Sortieren mit dem 6502 . . . 67
KIM decodiert Morsezeichen 68
RAM-Testprogramm für den Z-80-Kit 69
6502-Rechenroutinen 70
Textausgabe in Maschinensprache (Z-80) 72
6502-Disassembler 73
Video-Ausgabe beim AIM-65 74
Pseudo-Reset beim PET-2001 75
ASCII-Tastatur steuert KIM-1 76
Autostart-Einrichtung für AIM-65 und PC-100 77
6502 simuliert neue Adressierungsart 78
RTTY mit dem AIM-65 – jetzt mit Drucker 80
Eine Morseschreibmaschine (KIM) 82
AIM-65 decodiert Morsezeichen 83
Menue – Wie man Programme auch ohne Bandzählwerk wiederfindet (AIM-65) 84

Anm.: Alle AIM-65-Programme sind ohne Änderung auch auf dem PC-100 (Siemens) lauffähig.

Impressum

1980, 2., überarbeitete Auflage. ISSN 0172-2778.

© Franzis-Verlag GmbH, Karlstraße 37, 8000 München 2.

Bearbeitet von der Redaktion der Zeitschrift FUNKSCHAU. Für den Text verantwortlich: Herwig Feichtinger.

Sämtliche Rechte – besonders das Übersetzungsrecht – an Test und Bildern vorbehalten. Fotomechanische Vervielfältigungen nur mit Genehmigung des Verlages. Jeder Nachdruck, auch auszugsweise, und jede Wiedergabe der Abbildungen, auch in verändertem Zustand, sind verboten.

Druck: Franzis-Druck GmbH, Karlstraße 35, 8000 München 2.
ZV-Art.-Nr. 31012
F/ZV/780/288/7'

Wünschenswert ist ein Programm, das *alle* bei einer Basisumwandlung anstehenden Aufgaben erledigt: Beliebige Wahl der Basis von 2...99 999; exakte Hin- und Rückwandlung von bis zu zehn Vor- und neun Nachkommastellen. Ein solches Programm wird hier für die kleinsten programmierbaren Taschenrechner, HP-25 und 33E, vorgestellt. Für die größeren HP-Taschenrechner (19C/29C, 41C, 67/97) sind Labels als Sprungziele einzufügen.

Werner Frangen

Allround-Programm für die Basisumwandlung

Bei einer Basisumwandlung hat man es mit zwei verschiedenen Basen, ganzzahlig $\geqq 2$, zu tun. Eine davon, die *fremde Basis* g, ist keine Zehnerpotenz. Sie möge aus r Dezimalziffern bestehen ($r = 1, ..., 5$). Die andere sei die nächstgrößere *Zehnerpotenz*, $h = 10^r$.

Gegeben sei ein beliebiger Dezimalbruch $c = a.b \geqq 0$ mit bis zu 10 Vor- und s Nachkommastellen ($s = 10-r$). Das Programm wandelt ihn um in einen *fremden Systembruch* $f = d.e$ zur Basis g; und umgekehrt. Bei der Rückwandlung wird die Ausgangszahl wiederhergestellt (Probe!); der unvermeidliche maschinelle Rundungsfehler wird von r Schutzstellen aufgefangen.

Der fremde Systembruch hat im allgemeinen *unendlich viele* Nachkommastellen (wenn nicht zufällig g aus den gleichen Primteilern wie 10 zusammengesetzt ist, also $g = 2^x \cdot 5^y$, x und $y \geqq 1$). Das Programm errechnet beliebig viele Nachkommastellen. Praktisch aber wird man abbrechen, wenn die zur Rückwandlung erforderliche Genauigkeit erreicht ist. Dies gelingt mit höchstens *zwei Programmstarts*. Die endlich vielen Vorkommastellen werden dabei immer exakt berechnet.

Jede „Stelle" f_i des fremden Systembruchs f wird (ebenso wie die Basis g) mit r Dezimalziffern geschrieben ($0 \leqq f_i < g$). Faßt man die Ziffernkette f als Dezimalbruch auf, so ist

$$f = d.e = f_{m-1}...f_1f_0.f_{-1}f_{-2}...f_{-n} = \sum_{i=-n}^{m-1} f_i \cdot h^i. \qquad (1)$$

Bei der Rückwandlung wird daraus der eigentlich gemeinte Zahlenwert,

$$c \doteq a.b \doteq \sum_{i=-n}^{m-1} f_i \cdot g^i, \qquad (2)$$

berechnet. Da wir mit endlich vielen, n, Nachkommastellen abbrechen, begehen wir einen *Abbrechfehler* $-\varepsilon$ gegenüber dem ursprünglichen Dezimalbruch:

$$c = a.b = \sum_{i=-s}^{9} c_i \cdot h^i;\ c = \acute{c} + \varepsilon. \qquad (3)$$

Hierin ist jedes c_i eine „Stelle" zur Basis h ($0 \leqq c_i < h$); d. h. jedes c_i besteht aus r Dezimalziffern. Für den Abbrechfehler gilt:

$$0 \leqq \varepsilon \leqq g^{-n}. \qquad (4)$$

Um Fehler bei der Rückwandlung zu vermeiden, wird man

$$g^{-n} < \frac{1}{2} \cdot 10^{-10} \qquad (5)$$

verlangen. Diese Forderung wird gemäß Tab. 1 erfüllt.

Je kleiner die fremde Basis g ist, um so mehr Stellen sind bei gleicher Genauigkeit erforderlich; d. h. eine zehnziffrige Dezimalzahl erfordert bezüglich der Basis g im allgemeinen mehr als zehn Ziffern.

Jeder Programmstart liefert eine Zifferngruppe vor und eine nach dem Komma. Eine Zifferngruppe enthält bis zu zehn Ziffern. Jede Zifferngruppe besteht aus einer festen Anzahl v von „Stellen" zur Basis g oder h. Die Länge L einer Zifferngruppe (in Ziffern ausgedrückt) muß also ein ganzzahliges Vielfaches der Stellenlänge r sein:

$$L = v \cdot r \leqq 10. \qquad (6)$$

Je nach der Länge r der fremden Basis g ist somit eine *Fallunterscheidung* zu treffen (*Tabelle 1*).

Um eine ganze Zahl $< 10^{10}$ bezüglich der Basis g darzustellen, genügen demnach zwei Zifferngruppen (zwei Programmstarts). Ein einziger Programmstart reicht aus, wenn der Vorkommateil $< g^{-n}$ ist und beim Nachkommateil ein Fehler $\leqq g^{-v}$ toleriert wird.

Tabelle 2 Oktal ⇄ Binär

0	=	000
1	=	001
2	=	010
3	=	011
4	=	100
5	=	101
6	=	110
7	=	111

Tabelle 3 Nonal ⇄ Ternär

0	=	00
1	=	01
2	=	02
3	=	10
3	=	11
5	=	12
6	=	20
7	=	21
8	=	22

Tabelle 1. Fallunterscheidung nach der Länge der fremden Basis

Länge der fremden Basis g: r	1			2	3	4	5
Länge einer Zifferngruppe: L	10			10	9	8	10
Anzahl Stellen zur Basis g: v	10			5	3	2	2
Kleinste fremde Basen:	2	3	4	11	101	1001	10001
Größte Zahl mit zwei Zifferngruppen = 2 . v Stellen zur Basis g: $g^{2 \cdot v}-1$	$\approx 10^6$	$\approx 3 \cdot 10^9$	$> 10^{10}$				
Abbrechfehler nach zwei Zifferngruppen: $\varepsilon \leqq g^{-2 \cdot v}$	$<10^{-6}$	$<\frac{1}{3} \cdot 10^{-9}$	$<\frac{1}{2} \cdot 10^{-10}$				

Tabelle 4. Speicherbelegung und Programm
(I. Hinwandlung; II. Rückwandlung)

	I.	II.	
R0:	q^i		Quellbasispotenz;
R1:	a;	d_j	Vorkommateil oder -zifferngruppe } des Arguments
R2:	.b;	$.e_j$	Nachkommateil oder -zifferngruppe } (j = 1, 2);
R3:	q		Quellbasis;
R4:	d_j;	a	Vorkommazifferngruppe oder -teil } des Ergeb-
R5:	$.e_j$;	.b´	Nachkommazifferngruppe oder -teil } nisses;
R6:	z		Zielbasis;
R7:	w_i'		Durchlaufzähler; stoppt, wenn < 0.

Die eingerahmten Größen werden vor dem Programmstart manuell geladen.

			Anzeige/Kommentar
→	01	CLX	Vorbereitung:
	02	STO 4	$s_0 = 0.$
	03	STO 5	$t_0 = 0.$
	04	EEX	
	05	STO 0	$q^0 = 1.$
→	06	EEX	
	07	1	$w_0 = 10.$
●	08	RCL 6	Schleife:
	09	LOG	log z.
	10	–	w_i'.
	11	x < 0	(Fertig?)
	12	GTO 45	
	13	STO 7	
	14	RCL 6	z.
	15	RCL 1	p_{i-1}.
	16	x = 0	(Vorkommateil
	17	GTO 31	fertig?)
	18	RCL 1	p_{i-1}.
	19	RCL 6	z.
	20	÷	p_{i-1}/z.
	21	INT	p_i'.
	22	RCL 6	z.
	23	*	$p_i' \cdot z$.
	24	–	f_{i-1}.
	25	x < 0	(Korrektur nötig?)
	26	+	f_{i-1}.
	27	STO-1	
	28	RCL 0	q^{i-1}.
	29	*	$q^{i-1} \cdot f_{i-1}$.
	30	STO+4	R4: s_i.
●	31	RCL 6	
	32	STO÷1	R1: p_i.
	33	STO*2	R2: $q_{i-1} \cdot z$.
	34	RCL 3	q
	35	STO*0	R0: q^i.
	36	RCL 2	
	37	INT	f_{-i}.
	38	STO-2	R2 : q_i.
	39	RCL 0	
	40	÷	f_{-i}/q^i.
	41	STO+5	R5: t_i.
	42	RCL 7	w_i'.
	43	INT	w_i.
	44	GTO 08	Rücksprung.
●	45	RCL 4	d_1.
	46	RCL 5	$.e_1$.
	47	EEX	
	48	1	10
	49	*	$.e_1 \cdot 10$.

Die (bei HP-25 und 33E unterschiedlichen) Prefixtasten *f* und *g* sind weggelassen. * bedeutet die Multiplikation, ● Sprungziele, → Startadressen. Der Kommentar wird in der Programmanalyse (s. u.) näher erläutert.

Tabelle 5. Benutzeranweisung

Nr.	Anweisung	Daten	Schalter Tasten	Anzeige
1	Programm eintasten. Programmanfang.		RUN GTO 00	
2	Wahlweise → I. oder II.			
	I. *Hinwandlung aus dem Dezimalsystem*			
3	Zielbasis wählen (r-ziffrig, r ≦ 5)	g	STO 6	
	Quellbasis $h = 10^r$	h	STO 3 FIX	
	Anzeigeformat: l = L–1 (L nach Tab. 1).	l		
4	Vorkommateil des Arguments, $0 \leqq a < 10^{10}$.	a	STO 1	
	Höchstens s = 10–r Nachkommastellen.	.b	STO 2	
5	1. Programmstart; ergibt L NKSt.[4])		R/S	$.e_1 \cdot 10$
	(alle angezeigten St. als NKSt. notieren!)			
	und L VKSt. (führende Nullen ergänzen!)		x→y	d_1
6	2. Programmstart: L NKSt. (hinten an-		R/S	$.e_2 \cdot 10$
	fügen) und L VKSt. (vorne anfügen).[3,4])		x→y	d_2
7	*Ergebnis:* $f = d_2 d_1 . e_1 e_2$. Für größere			
	Genauigkeit wiederhole → 6 beliebig oft.			
	Neue Hinwandlung: → 4;			
	Basisänderung: → 2.			
	II. *Rückwandlung in das Dezimalsystem*			
8	Quellbasis wählen (r-ziffrig, r ≦ 5).[1])	g	STO 3	
	Zielbasis $h = 10$.	h	STO 6 FIX	
	Anzeigeformat: s = 10–r	s		
9	Argument: L innere VKSt. (L nach Tab. 1);	d_1	STO 1	
	L innere NKSt.	$.e_1$	STO 2	
	1. Programmstart		R/S	
10	Argument: L äußere VKSt.	d_2	STO 1	
	L äußere NKSt.	$.e_2$	STO 2	
	2. Programmstart (nicht R/S!).[2,3])		GSB 06	
11	Die letzte angezeigte Stelle von b' ist			.b' · 10
	als Schutzstelle zu betrachten. Die vor-			
	angehenden s Stellen als NKSt. notieren![4])			
	Von a nur die VKSt. notieren!		x→y	a
	Ergebnis: c' = a.b' (unbrauchbar, falls $\geqq 10^{10}$).			
	Neue Rückwandlung: → 9;			
	Basisänderung: → 2.			

[1]) Beim HP-25 nur r ≦ 4, weil er fälschlich $\log 10^5 > 5$ berechnet. Abhilfe: ↑ 5 in Programmschritt 08...09.
[2]) Beim HP-25: *GTO 06 R/S*.
[3]) Für g = 2 oder 3 wäre diese Anweisung noch zweimal bzw. einmal zu wiederholen.
[4]) Die Verzehnfachung macht alle Nachkommastellen ablesbar.

Tabelle 6. Beispiele mit Rechenzeiten

Nr.	Dezimalbruch		Fremder Systembruch (Basis)	Rechenzeit [s] HP-25	33E
1	1234 56789.98765 4321	=	$2772\ 66780.88000\ 0000_{(9)}$	24	43
2	99999 99999.99999 9999	=	11'24027 61777.		
			$.77777\ 77776'73217\ 50137_{(8)}$	46	70
3	99999 99999.99999 999	=	04 02 07 01 08'01 05 00 01 00.		
			$.10\ 10\ 10\ 10\ 10'10\ 10\ 08\ 09\ 04_{(11)}$	27	45
4	99999 99999.99999 99	=	096 009'091 004 099.		
			$.100\ 100\ 100'090\ 059\ 099_{(101)}$	18	28
5	99999 99999.99999 9	=	9 0971'0029 0990.		
			$.1000\ 0999'0998\ 0997_{(1001)}$	12	20
6	c = .74983 4881	=	$.20202\ 01212'22222\ 22202'10120_{(3)}$	54	84
	c' = .74983 48815	=			

Ausnahme: *Binärzahlen* (g = 2) würden vier, *Ternärzahlen* (g = 3) drei Zifferngruppen erfordern. Um die Rechenzeit abzukürzen, wandelt man statt dessen besser in *Oktalzahlen* (g = 8) bzw. *Nonalzahlen* (g = 9) um. Zur Umrechnung benutzt man die *Tabellen 2* und *3*.

Von den beiden Basen g und h dient jeweils eine als *Zielbasis* z und die andere als *Quellbasis* q. Für die Rückwandlung werden Ziel- und Quellbasis vertauscht. Das Rechenverfahren ist das gleiche. Darin besteht der Witz des Programms, das in *Tabelle 4* wiedergegeben ist. Die Benutzung geht aus *Tabelle 5* hervor, und *Tabelle 6* zeigt einige Beispiele mit Angabe der Rechenzeiten.

Tabelle 7. Speicherbelegung und Programm (I. Hinwandlung; II. Rückwandlung)

	I.	II.		
R0:	q		Quellbasispotenz;	
R1:	c;	d_j	Argument oder Zifferngruppe des Arguments (j = 1, 2);	
R2:	r		Länge der Fremdbasis g;	
R3:	q		Quellbasis;	
R4:	d_2;	c	Vordere Zifferngruppe des Ergebnisses (nach zweimaligem Programmstart) oder Ergebnis;	
R5:	d_1		Hintere Zifferngruppe des Ergebnisses (nach zweimaligem Programmstart);	
R6:	z		Zielbasis;	
R7:	w_i		Durchlaufzähler; stoppt, wenn < 0.	

	00	10	20	30	40
0		x < 0	INT	RCL 6	GTO 48
1	→ EEX	GTO 49	RCL 6	STO÷1	→ STO 6
2	STO 0	STO 7	*	RCL 3	LOG
3	RCL 4	RCL 6	–	STO*0	EEX
4	STO-4	RCL 1	x < 0	RCL 7	+
5	STO 5	x = 0	+	GTO 08	INT
6	→ EEX	GTO 49	STO-1	→ RCL 6	STO 2
7	1	RCL 1	RCL 0	RCL 3	10
8	● RCL 2	RCL 6	*	STO 6	● STO 3
9	–	÷	STO+4	R↓	● RCL 4

Tabelle 8. Benutzeranweisung

Nr.	Anweisung	Daten	Schalter Tasten	Anzeige
1	Programm eintasten. Anzeigeformat.		RUN FIX 0	
2	Wähle Fremdbasis (r-ziffrig, r ≦ 5).[1])	g	GSB 41	
3	Wahlweise → 4 oder → 7, 8.			
	I. Hinwandlung aus dem Dezimalsystem			
4	Argumentzahl ($0 < c < 10^{10}$)	c	STO 1	
	1. Programmstart liefert L Ziffern		R/S	d_1
	(L s. Tab. 1); führende Nullen ergänzen!			
5	2. Programmstart liefert L Ziffern.[2,3])		R/S	d_2
6	*Ergebnis:* $c = d_2 d_{1(g)}$.			
	Neue Rechnung: → 3.			
	Änderung der Fremdbasis: → 2.			
7	Vertauschung von Ziel- und Quellbasis.[1])		GSB 36	
	II. Rückwandlung in das Dezimalsystem			
8	Argument: L niederwertige Ziffern.[4])	d_1	STO 1 R/S	
9	Argument: L höherwertige Ziffern.)	d_2	STO 1 GSB 06	
10	*Ergebnis:* c (≧ 10^{10} unbrauchbar).			c
	Neue Rechnung: Wahlweise → 8 oder → 7, 4.			
	Änderung der Fremdbasis: → 2.			

[1]) Beim HP-25 drücke: GTO nn R/S anstelle von GSB nn!
[2]) Überflüssig, wenn $c < g^v$ (v. s. Tab. 1)
[3]) Im Ausnahmefall g = 2 oder 3 wäre diese Anweisung noch zweimal bzw. einmal zu wiederholen.
[4]) Die Zifferngruppen d_1 bzw. d_2 sind aus Register R5 zu entnehmen (mit RCL 5), wenn sie zuletzt durch Hinwandlung erzeugt wurden.
[5]) Überflüssig, wenn $d_2 = 0$.

Das 1. Beispiel eignet sich zur Prüfung, ob das Programm richtig geladen wurde (nur 1 Programmstart). Die Beispiele 2...5 schöpfen die Rechenkapazität des Programms voll aus (2 Programmstarts). Das Beispiel 6 (3 Programmstarts) zeigt in der Schutzstelle den größten Fehler, der bei der Rückwandlung beobachtet wurde.

Die Rechenzeit verringert sich ungefähr um die Hälfte (gegenüber Nr. 2...5), wenn nur 1 Programmstart erforderlich ist, und um ein Drittel, wenn der Vorkommateil Null ist.

Programmanalyse: Für solche Leser, die das Verfahren auf ihren Computer übertragen wollen, sind ein paar zusätzliche Erläuterungen am Platze. Alles Wesentliche ergibt sich aus der Betrachtung einer Hinwandlung nach dem 1. Programmstart. Bis zum ersten Stopp wird die Programmschleife v-mal durchlaufen. Beim i-ten Durchlauf (i = 1...v) werden die folgenden Größen berechnet; die Startwerte erhalten den Index i = 0:

In R1: p_i (Vorkomma-Argument im Abbau); $p_0 = a$;
in R2: q_i (Nachkomma-Argument im Abbau); $q_0 = .b$;

in R4: s_i (Vorkomma-Ergebnis im Aufbau); $s_0 = 0$;
in R5: t_i (Nachkomma-Ergebnis im Aufbau); $t_0 = 0$.

Gerechnet wird nach den Formeln:

Vorkommateil.

(7) Abbau:
$p_i = \mathrm{INT}(p_{i-1}/z)$; $\quad p_{2:v} = 0$.

(8) Vorkommastelle:
$f_{i-1} = p_{i-1} - z : p_i$.

(9) Aufbau:
$s_i = s_{i-1} + f_{i-1} : q$; $\quad s_v = d_1$.

Nachkommateil.

(10) Abbau:
$q_i = \mathrm{FRAC}(q_{i-1} : z)$.

(11) Nachkommastelle:
$f_{-i} = \mathrm{INT}(q_{i-1} : z)$.

(12) Aufbau:
$t_i = t_{i-1} + f_{-i}/q$; $\quad t_v = .e_1$.

Die Formel (7) erfordert eine Korrektur, die in den Programmschritten 14, 25...26 erfolgt. Der ganzzahlige Quotient INT(p/z) wird nämlich auf zehnstellig arbeitenden Rechnern für Dividenden $p > 2 \cdot 10^9$ manchmal um 1 zu groß berechnet; z. B. für p = 2 000 000 001 und z = 2. Aus dem infolgedessen falschen Divisionsrest $f' < 0$ macht die Korrektur den richtigen Rest $f = f' + z$.

Die Korrektur versagt lediglich für das Wertepaar $p_0 = a = 99999\ 99999$, $z = 6$. Der Fehler wird bei der Rückwandlung an dem Ergebnis $c' = 10^{10}$ erkennbar.

Im übrigen arbeiten die Formeln exakt. Nur bei der Rückwandlung macht die Formel (12) im Nachkommateil kleine Rundungsfehler, die aber von den Schutzstellen aufgefangen werden.

Die Durchlaufzählung wird mit dem Startwert $w_0 = 10$ begonnen. Im i-ten Durchlauf werden die Größen $w'_i = w_{i-1} - \log z$ und

$$w_i = \mathrm{INT}(w'_i) = w_{i-1} - r = w_0 - i \cdot r$$

gebildet. Am Anfang des nächsten Durchlaufs wird geprüft, ob $w_{i+1} = w_i - \log z < 0$ ist; wenn ja, wird gestoppt. Auf diese Weise werden genau v Durchläufe erzielt.

Die Rechenzeit läßt sich noch verbessern, wenn man nicht mit den Befehlen knausern muß.

Ganzzahlige Basisumwandlung

Sind nur ganze Zahlen umzuwandeln, zum Beispiel die Adressen eines Hobbycomputers, dann läßt sich ein doppelt so schnelles Programm für den HP-25 und HP-33E angeben wie das vorstehende. Die Bedienung ist komfortabler, das Rechenverfahren (für den Vorkommateil) bleibt das gleiche. Das Programm findet sich in *Tabelle 7* und die Benutzeranweisung in *Tabelle 8*. ■

Hex-BCD-Umwandlung mit dem Z-80

In dem RPB-Band 173, „Anwendungsbeispiele für den 6502", findet sich ein auffallend kurzes Programm zur Umwandlung einer Binär- in eine Dezimalzahl. Die zugrundeliegende Methode läßt sich natürlich auch auf andere Prozessortypen übertragen, wenn man den DAA-Befehl zu Hilfe nimmt.

Das im *Bild* aufgelistete Z-80-Programm wandelt eine 16-bit-Binärzahl im HL-Register in eine Dezimalzahl um. Das Ergebnis (maximal 65536) steht im gepackten BCD-Format in den Registern C, D und E. Statt HL kann auch eines der Indexregister IX und IY verwendet werden; der Befehl ADD HL,HL (der als Linksschiebe-Operation anzusehen ist) muß dann durch ADD IX,IX bzw. ADD IY,IY ersetzt werden.

Klaus Betke

```
                                  HEXDEA                                PAGE   1
LOC   OBJ CODE M STMT SOURCE STATEMENT                                   ASM 5.8
                      1  ;        UMWANDLUNG EINER 16 BIT-DUALZAHL IN HL
                      2  ;        IN EINE BCD-ZAHL. DAS MAXIMAL 5-STELLIGE
                      3  ;        ERGEBNIS STEHT IN DEN REGISTERN C,D UND E.
                      4
                      5           GLOBAL HEXDEA
                      6
                      7
                      8  HEXDEA:
0000   AF             9           XOR A
0001   57            10           LD D,A
0002   0610          11           LD B,16
                     12  CONV:
0004   29            13           ADD HL,HL
0005   8F            14           ADC A,A
0006   27            15           DAA
0007   5F            16           LD E,A
0008   7A            17           LD A,D
0009   8F            18           ADC A,A
000A   27            19           DAA
000B   57            20           LD D,A
000C   CB11          21           RL C
000E   7B            22           LD A,E
000F   10F3          23           DJNZ CONV
0011   C9            24           RET
```

Das hier vorgestellte Programm für den Rechner TI-59 ermittelt den Strahlungswiderstand von Monopolen und Dipolen. Es stellt daher eine wertvolle Hilfe bei der Errechnung von Anpaßschaltungen für „unübliche" Antennenlängen dar.

Ing. (grad.) Fred H. Heller

TI-59 berechnet Strahlungswiderstand

Das Programm *(Bild)* arbeitet nach der Formel

$$R_s = 60 \cdot \int_0^{\pi/2} L^2(\varphi) \cos \varphi \, d\varphi$$

mit R_s = Strahlungswiderstand, L = vertikales Strahlungsdiagramm und φ = Elevationswinkel. Da der Rechner TI-59 leider nicht auf eine Integralsinus-Tabelle zurückgreifen kann, ist ein Unterprogramm vorhanden, das ihm die Werte für Integralsinus und -cosinus liefert; es belegt übrigens die Speicher 4, 7, 10, 11, 12, 13, 15, 16, 18 und die Programmarken A' und B'.

Für den Dipol gilt, daß sich der Strahlungswiderstand aus der Reihenschaltung zweier entsprechend halb so langer Monopole ergibt. Durch Druck auf die Label-Taste D werden zwei Flags gesetzt, die eine entsprechende Programmänderung bewerkstelligen; nach Abschluß der Dipolberechnung schalten sich die Flags automatisch wieder in die Monopol-Berechnungsart zurück.

Die Eingabemöglichkeiten sind recht komfortabel. Will man z. B. für 27 MHz entsprechend 11 m Wellenlänge den Strahlungswiderstand eines 5,5 m langen Monopols errechnen, also für einen Halbwellen-Strahler, so kann man entweder 11 mit B oder 27 mit C eingeben, 5,5 dagegen mit A *(Tabelle)*.

```
000  91 R/S     041  19  19     082  55  -      123  65  ×      164  00  0      240  76 LBL
001  76 LBL     042  32 X:T     083  02  2      124  43 RCL     165  32 X:T     241  16 A'
002  11  A      043  08  8      084  95  =      125  22  22     166  77  GE     242  42 STO
003  42 STO     044  65  ×      085  23 LNX     126  54  )      167  00  00     243  10  10
004  14  14     045  89  π      086  42 STO     127  85  +      168  05  05     244  42 STO
005  91 R/S     046  95  =      087  25  25     128  01  1      169  01  1      245  15  15
006  76 LBL     047  22 INV     088  25 CLR     129  05  5      170  94 +/-     246  65  ×
007  12  B      048  77  GE     089  03  3      130  65  ×      171  34 FIX     247  02  2
008  85  +/-    049  01  01     090  00  0      131  53  (      172  61 GTO     248  85  +
009  65  ×      050  69  69     091  65  ×      132  43 RCL     173  00  00     249  02  2
010  43 RCL     051  71 SBR     092  53  (      133  17  17     174  00  00     250  95  =
011  14  14     052  16 A'      093  00  0      134  39 COS     175  55  ÷      251  59 INT
012  95  =      053  42 STO     094  93  .      135  54  )      176  02  2      252  42 STO
013  61 GTO     054  20  20     095  05  5      136  65  ×      177  95  =      253  11  11
014  00  00     055  43 RCL     096  07  7      137  53  (      178  86 STF     254  01  1
015  26  26     056  19  19     097  07  7      138  00  0      179  02  02     255  42 STO
016  76 LBL     057  71 SBR     098  02  2      139  93  .      180  61 GTO     256  12  12
017  13  C      058  17 B'      099  01  1      140  05  5      181  00  00     257  42 STO
018  55  ÷      059  42 STO     100  06  6      141  07  7      182  30  30     258  13  13
019  03  3      060  21  21     101  85  +      142  07  7      183  65  ×      259  42 STO
020  00  0      061  43 RCL     102  43 RCL     143  02  2      184  02  2      260  04  04
021  00  0      062  17  17     103  24  24     144  01  1      185  95  =      261  42 STO
022  65  ×      063  71 SBR     104  75  -      145  06  6      186  32 X:T     262  16  16
023  43 RCL     064  16 A'      105  43 RCL     146  85  +      187  00  0      263  42 STO
024  14  14     065  42 STO     106  23  23     147  43 RCL     188  32 X:T     264  18  18
025  95  =      066  22  22     107  54  )      148  25  25     189  22 INV     265  61 GTO
026  87 IFF     067  43 RCL     108  85  +      149  85  +      190  77  GE     266  03  03
027  01  01     068  17  17     109  01  1      150  43 RCL     191  01  01     267  01  01
028  01  01     069  71 SBR     110  05  5      151  21  21     192  69  69     268  76 LBL
029  75  75     070  17 B'      111  65  ×      152  75  -      193  81 RST     269  17 B'
030  65  ×      071  42 STO     112  53  (      153  02  2      194  76 LBL     270  42 STO
031  04  4      072  23  23     113  43 RCL     154  65  ×      195  14  D      271  10  10
032  65  ×      073  25 CLR     114  17  17     155  43 RCL     196  86 STF     272  23 LNX
033  89  π      074  70 RAD     115  38 SIN     156  23  23     197  01  01     273  85  +
034  95  =      075  43 RCL     116  54  )      157  54  )      198  61 GTO     274  00  0
035  42 STO     076  17  17     117  65  ×      158  95  =      199  00  00     275  93  .
036  17  17     077  23 LNX     118  53  (      159  87 IFF     200  05  05     276  05  5
037  65  ×      078  42 STO     119  43 RCL     160  02  02
038  02  2      079  24  24     120  20  20     161  01  01
039  95  =      080  43 RCL     121  75  -      162  83  83
040  42 STO     081  17  17     122  02  2      163  32 X:T
```

(Fortsetzung des Programms auf der nächsten Seite)

```
277  07   7  | 296  01   1  | 315  94  +/- | 334  10  10  | 353  00   0  | 372  65   ×
278  07   7  | 297  42  STO | 316  49  PRD | 335  45  YX  | 354  42  STO | 373  01   1
279  02   2  | 298  16  16  | 317  16  16  | 336  43  RCL | 355  04  04  | 374  00   0
280  01   1  | 299  42  STO | 318  69  OP  | 337  12  12  | 356  42  STO | 375  00   0
281  06   6  | 300  18  18  | 319  27  27  | 338  55   ÷  | 357  07  07  | 376  00   0
282  95   =  | 301  43  RCL | 320  69  OP  | 339  43  RCL | 358  42  STO | 377  00   0
283  42  STO | 302  11  11  | 321  24  24  | 340  13  13  | 359  10  10  | 378  00   0
284  15  15  | 303  32  X:T | 322  43  RCL | 341  55   ÷  | 360  42  STO | 379  95   =
285  43  RCL | 304  43  RCL | 323  04  04  | 342  43  RCL | 361  11  11  | 380  59  INT
286  10  10  | 305  07  07  | 324  49  PRD | 343  18  18  | 362  42  STO | 381  55   ÷
287  65   ×  | 306  77  GE  | 325  18  18  | 344  65   ×  | 363  12  12  | 382  01   1
288  03   3  | 307  03  03  | 326  69  OP  | 345  43  RCL | 364  42  STO | 383  00   0
289  55   ÷  | 308  53  53  | 327  24  24  | 346  16  16  | 365  13  13  | 384  00   0
290  02   2  | 309  02   2  | 328  43  RCL | 347  95   =  | 366  42  STO | 385  00   0
291  85   +  | 310  44  SUM | 329  04  04  | 348  44  SUM | 367  16  16  | 386  00   0
292  03   3  | 311  12  12  | 330  49  PRD | 349  15  15  | 368  42  STO | 387  00   0
293  95   =  | 312  44  SUM | 331  18  18  | 350  61  GTO | 369  18  18  | 388  95   =
294  42  STO | 313  13  13  | 332  25  CLR | 351  03  03  | 370  43  RCL | 389  92  RTN
295  11  11  | 314  01   1  | 333  43  RCL | 352  01  01  | 371  15  15  |
```

Das TI-59-Programm. Die Programmschritte 201...239 sind hier freigelassen; bei 240 beginnt die Routine zur Errechnung der Integrale

Eingabe der Werte in den TI-59

Schritt	Prozedur	Taste	Anzeige
1	Antennenlänge eingeben	A	l
2	Nur bei Dipol: Flags setzen	D	l
3	Wellenlänge eingeben	B	R_s
3	oder Frequenz eingeben	C	R_s
4	Für andere Frequenzen bei der gleichen Ant. weiter bei Schritt 2		
Berechnung des Integralsinus/-cosinus:			
5	X eingeben	A'	Si (X)
6	X eingeben	B'	Ci (X)

Zur Berechnung des Strahlungswiderstandes bei diesem Beispiel (Ergebnis: 99.5 beim Monopol und 73.1 beim Dipol) braucht der Rechner rund fünf Minuten, was auf die recht langwierige Integralsinus- und -cosinus-Reihenentwicklung zurückzuführen ist. Für kleinere Antennenlängen im Verhältnis zur Wellenlänge braucht der TI-59 deutlich weniger Zeit, da die Schleifenzahl zur Anpassung an die erforderliche Genauigkeit automatisch geändert wird.

Literatur

Meinke/Gundlach: Taschenbuch der Hochfrequenztechnik, 3. Aufl., S. 526. ■

Berechnung von Yagi-Antennen

Für den Fernseh- und UKW-Rundfunkempfang verwendet man ebenso wie im VHF-UHF-Amateurfunk vorwiegend Mehrelement-Richtantennen, die aus einem gespeisten Dipol, mehreren Direktoren und einem Reflektor bestehen. Eine solche Antenne wird auch – nach ihrem Erfinder – „Yagi-Antenne" genannt.

Das im *Bild* aufgelistete Basic-Programm für den PET-2001 erlaubt das Berechnen der mechanischen Abmessungen einer solchen Antenne für beliebige Frequenzen. Zu bedenken ist, daß hier ein eventueller Verkürzungsfaktor durch große Strahler- oder Elementdurchmesser nicht berücksichtigt wird. Die Ergebnisse sind jedoch für die Praxis hinreichend genau.

Im Listing scheinbar zusammenhanglos stehende Buchstaben oder Grafikzeichen entsprechen bestimmten Steuerzeichen des PET-2001, z. B. bedeutet ein „R" das Umschalten in den Reverse-Modus (weißer Hintergrund). Das Programm belegt ohne Variablen 1554 Bytes, mit Variablen 1611 Bytes.

Thomas Grüger

```
7 PRINT"♥"
8 POKE59409,52
10 PRINT"        Y A G I - A N T E N N E N"
11 PRINT"        ========================="
20 PRINT:PRINT:PRINT
30 PRINT"              ▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄▄"
40 PRINT"                      ▌"
50 PRINT"                      ▌"
60 PRINT"                      ▌"
70 PRINT"                      ▌"
80 PRINT"                      ▌"
90 PRINT"       R                    ▄▌"
100 PRINT"                     ▌"
110 PRINT"                     ▌"
120 PRINT"                     ▌"
130 PRINT"                     ▌"
140 PRINT"                     ▌"
150 PRINT"                     ▌"
160 PRINT"     R                      ▄▌"
170 PRINT
171 POKE59409,60
190 PRINT"FREQUENZ(KHZ)";
200 INPUTF:IFF<1THEN700
210 W=300000/F
220 S=.465*W:D=.44175*W:R=.48825*W
230 AD=.12*W:AR=.15*W:GOSUB500
235 PRINT"SQQQQQDIREKTOR=  <-";D;"M";"->"
240 PRINT"QQQQSTRAHLER=  <-";S;"M";"->"
250 PRINT"QQQQQREFLEKTOR= <-";R;"M";"->"
260 PRINT"SQQQQQQ"
270 PRINTTAB(28);"/\"
280 PRINTTAB(28);" |"
290 PRINTTAB(28);" |"
300 PRINTTAB(26);AD;"M"
310 PRINTTAB(28);" |"
320 PRINTTAB(28);"\/"
330 PRINTTAB(32);"/\"
340 PRINTTAB(32);" |"
350 PRINTTAB(32);" |"
360 PRINTTAB(30);AR;"M"
370 PRINTTAB(32);" |"
380 PRINTTAB(32);" |"
390 PRINTTAB(32);"\/"
400 GOTO600
500 D=D*100:D=INT(D):D=D/100
510 S=S*100:S=INT(S):S=S/100
520 R=R*100:R=INT(R):R=R/100
530 AD=AD*100:AD=INT(AD):AD=AD/100
540 AR=AR*100:AR=INT(AR):AR=AR/100
545 IFD=0ORS=0ORR=0ORAD=0ORAR=0THEN700
550 RETURN
600 PRINT"QQQNEUE BERECHNUNG ?  (J/N)"
610 GETZ$:IFZ$=""THEN610
620 IFZ$="J"THEN5
630 PRINT"♥":END
700 PRINT"♥FUER DIE FREQUENZ ";F;" KHZ"
710 PRINT"KANN KEINE ANTENNE BERECHNET WERDEN."
720 GOTO600
```

Für die Benutzer des programmierbaren Taschenrechners HP-67 wird ein Programm vorgestellt, mit dem sich das Nachschlagen in Lohn- bzw. Einkommensteuertabellen erübrigt. Die Programmbeschreibung ist jedoch so aufgebaut, daß auch der an der Steuerberechnung interessierte „Nicht-HP-67-Besitzer" sie als Anleitung zur Berechnung mit einfacheren Rechnern benutzen kann.

Bernd Sundermeier

Berechnung der Lohn- und Einkommensteuer mit dem HP-67

Das Programm berücksichtigt die seit 1. 1. 1979 geltenden Steuergesetze und erlaubt sowohl die Berechnung der Einkommensteuer nach Grund- und Splittingtarif als auch der monatlichen Lohnsteuer für verschiedene Steuerklassen unter Berücksichtigung der anzurechnenden Kinder.

Einkommensteuer laut Grundtarif

Den Kern des Gesamtprogramms bildet das Unterprogramm Label B, mit dem die jährliche Einkommensteuer gemäß § 32a des Einkommensteuergesetzes berechnet werden kann. Die Berechnung der Einkommensteuer EST erfolgt je nach Höhe des Jahreseinkommens mit Hilfe folgender Gleichungen:

für zu versteuernde Einkommen bis DM 3719:

$$EST = 0 \qquad (1)$$

für zu versteuernde Einkommen von DM 3720 bis DM 16 019:

$$EST = 0{,}22x - 812 \qquad (2)$$

für zu versteuernde Einkommen von DM 16 020 bis DM 47 999:

$$EST = (((10{,}86y - 154{,}42)y + 925)y + 2200)y + 2708 \qquad (3)$$

für zu versteuernde Einkommen von DM 48 000 bis DM 130 019:

$$EST = (((0{,}1\,z - 6{,}07)z + 109{,}95)z + 4800)z + 15\,298 \qquad (4)$$

für zu versteuernde Einkommen über DM 130 020:

$$EST = 0{,}56x - 13\,644 \qquad (5)$$

Das zu versteuernde Einkommen wird berechnet durch Abrundung des tatsächlichen Jahreseinkommens auf den nächsten durch 30 teilbaren Betrag bei Einkommen unter DM 48 000 und auf den nächsten durch 60 teilbaren Betrag bei Einkommen über DM 48 000. „x" ist das abgerundete zu versteuernde Einkommen; „y" ist ein Zehntausendstel des DM 16 000 übersteigenden und „z" ein Zehntausendstel des DM 48 000 übersteigenden Teils des abgerundeten Einkommens.

Die Berechnung der Einkommensteuer erfolgt nach Eingabe des Jahreseinkommens und Drücken der Taste „B". Der prinzipielle Aufbau des Programms ist dem Programmablaufplan in *Bild 1* zu entnehmen, das fertige Programm (bzw. Unterprogramm) ist im Listing von Schritt 136 bis 224 angegeben.

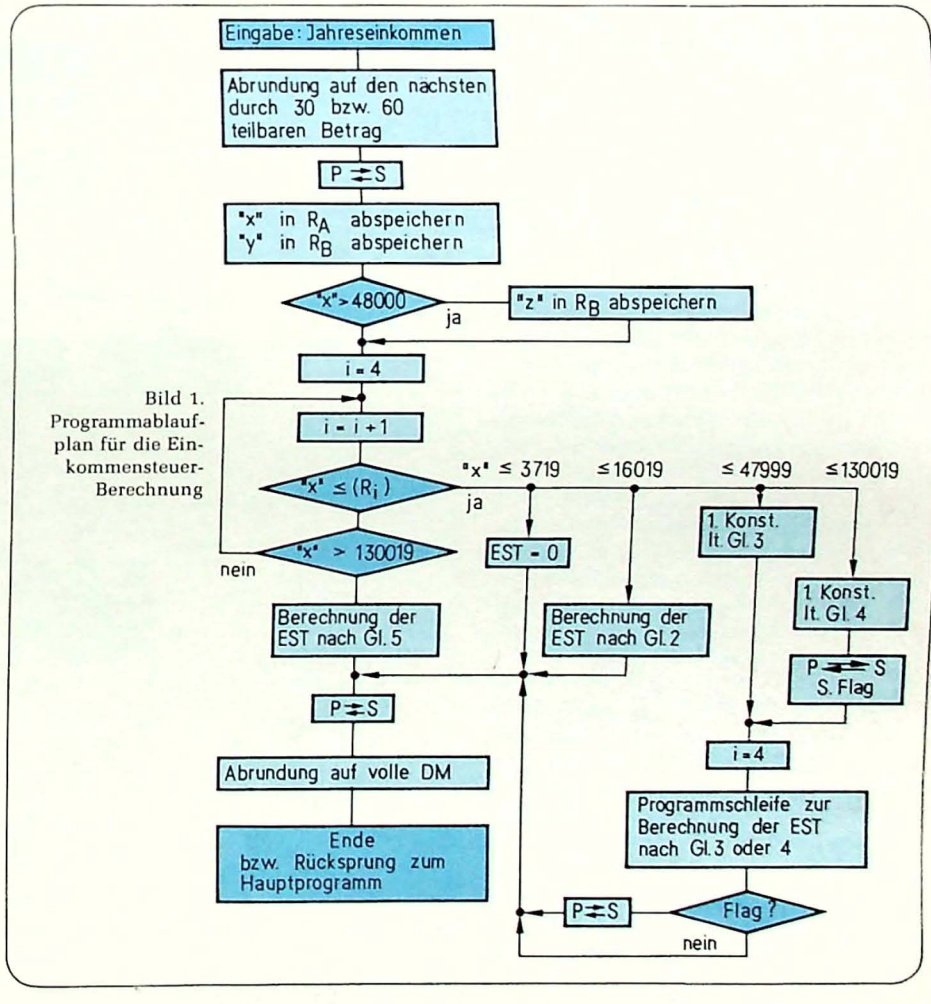


Bild 1. Programmablaufplan für die Einkommensteuer-Berechnung

(3) = 10,86 y^4 − 154,42 y^3 + 925 y^2 + 2200 y + 2708

(4) = 0,1 z^4 − 6,07 z^3 + 109,95 z^2 + 4800 z + 15298

Einkommensteuer laut Splittingtarif

Die Bestimmung der Einkommensteuer bei Ehegatten, die zusammen zur Einkommensteuer veranlagt werden, erfolgt anhand des sogenannten Splittingtarifs. Die Steuer laut Splittingtarif ist das Doppelte des Steuerbetrages, der sich für die Hälfte des gemeinsam zu versteuernden Einkommens nach dem oben erläuterten Verfahren ergibt. Das Programm zur Berechnung des Splittingtarifs ist also sehr einfach und erfordert nur wenige Programmschritte (siehe Label C, Schritte 126 bis 131 im Listing). Die eigentliche Bestimmung der Steuer erfolgt nämlich gemäß Grundtarif, so daß hierfür das Programm unter Label B als Unterprogramm aufgerufen werden kann. Die Berechnung der Einkommensteuer für zusammen veranlagte Ehegatten wird nach Eingabe des zu versteuernden Jahreseinkommens und Drücken der Taste „C" durchgeführt.

Lohnsteuer für Monatseinkommen

Die Berechnung der monatlichen Lohnsteuer ist aufgrund der verschiedenen Steuerklassen und Freibeträge recht kompliziert. Zum besseren Verständnis des Programms und um das Programm bei evtl. späteren Steuergesetzänderungen leichter an die neuen Verhältnisse anpassen zu können, ist in Bild 2 der prinzipielle Programmablauf, aus dem die einzelnen Rechenschritte in etwa hervorgehen, vereinfacht dargestellt.

Zu Beginn der Lohnsteuerberechnung muß der Monatsbetrag auf den in den amtlichen Monatslohnsteuertabellen vorgegebenen Betrag aufgerundet werden, da sonst bei einigen Werten durch die Rundung der Vorsorgepauschale und des Jahreseinkommens geringfügige Fehlberechnungen auftreten können. Die Aufrundung erfolgt auf den nächsten Betrag, der auf 1,99 oder 4,49 oder 6,99 oder 9,49 endet.

Je nach Steuerklasse und Anzahl der zu berücksichtigenden Kinder werden vor der Steuerberechnung die Freibeträge (FB) und die Vorsorgepauschale (VP) vom Jahreseinkommen abgezogen. Die Höhe der Freibeträge und die Ermittlung der Vorsorgepauschale sind in Tabelle 1 angegeben. Bei der Berechnung der Vorsorgepauschale ist ein Höchstbetrag von 9 % der gültigen Beitragsbemessungsgrenze der gesetzlichen Rentenversicherung zu berücksichtigen, der für 1979 DM 4320,– (9 % von DM 48 000) beträgt und in R_C gespeichert ist. Nach künftigen Erhöhungen der Beitragsbemessungsgrenze muß also der Inhalt von R_C entsprechend korrigiert werden.

Berechnung der Vorsorgepauschale und des Freibetrages (alle Beträge in DM)

Steuerklasse	1	2	3	4	5	6
Vorsorgepauschale	9 % vom Jahresgehalt (Monatsgehalt x 12)*–				–	
jedoch höchstens	2100	2100	4200	2100	–	–
zuzügl. pro Kind	600	600	600	300	–	–
zuzüglich	9 % vom Jahresgehalt (s. o.)				–	–
jedoch höchstens	1050	1050	2100	1050	–	–
zuzügl. pro Kind	300	300	300	150	–	–
Die zu berücksichtigende Vorsorgepauschale (VP) ergibt sich durch Abrundung auf den nächsten durch 30 teilbaren Betrag.						
Arbeitnehmerfreibetrag	480	480	480	480	480	–
Werbungskostenpauschale	564	564	564	564	564	–
Sonderausgabenpauschale	240	240	480	240	–	–
Haushaltsfreibetrag	–	840**	–	–	–	–
Tariffreibetrag	510	510	1020	510	–	–
Rundungsbetrag	–	–	–	–	–	24
Summe der Freibeträge (FB)	1794	2634**	2544	1794	1044	24

* höchstens jedoch 9 % vom Jahresbetrag der maßgebenden Beitragsbemessungsgrenze in der gesetzlichen Rentenversicherung der Angestellten.

** Bei Berücksichtigung von einem oder mehreren Kindern erhöhen sich die Beträge um 2160 DM.

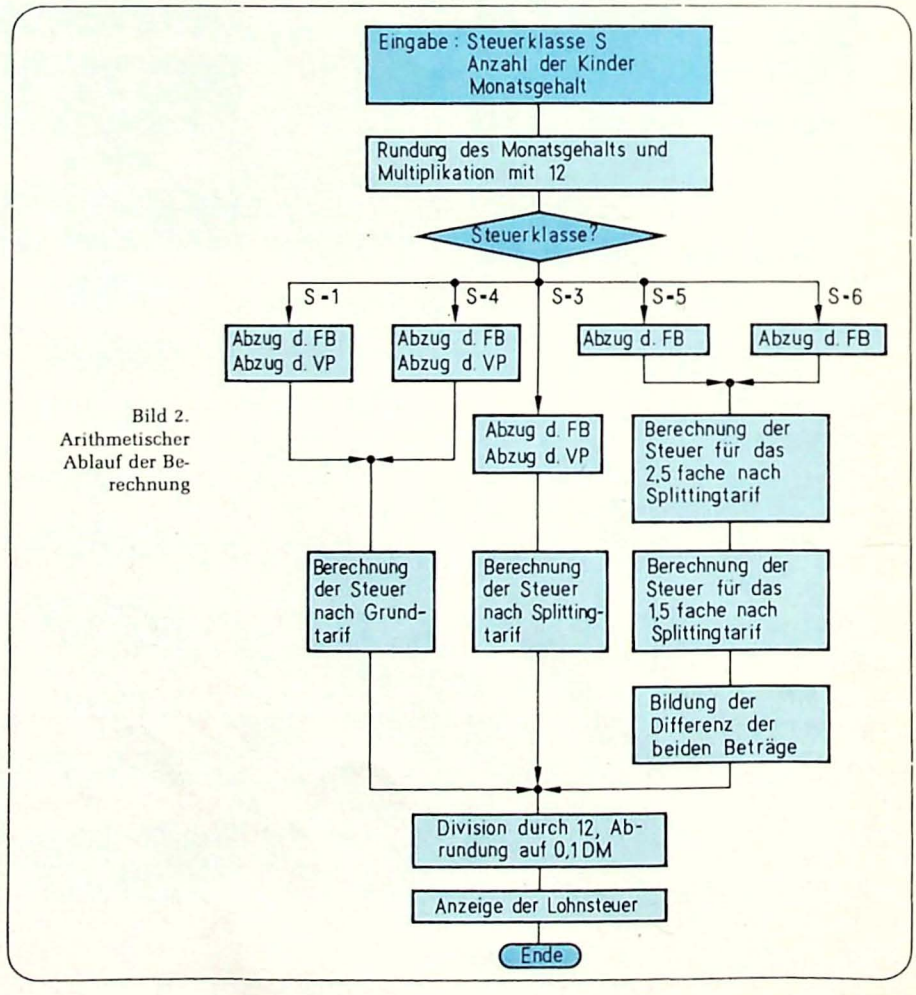


Bild 2. Arithmetischer Ablauf der Berechnung

```
001  LBL A      059  STO + 9    117  LBL 8      153  INT        190  -
002  .          060  GSB 7      118  1          154  x          191  GOTO 5
003  5          061  GSB C      119  .          155  F? 1       192  LBL 6'
004  +          062  GOTO D     120  5          156  RTN        193  RCL 2
005  .          063  LBL 5      121  x          157  LBL B'     194  x
006  4          064  RCL A      122  GOTO E     158  P≠S        195  8
007  x          065  RCL D      123  LBL 9      159  LEX        196  1
008  INT        066  GOTO 3'     124  +          160  4          197  2
009  .          067  LBL 6      125  GOTO E     161  :          198  -
010  4          068  RCL A      126  LBL C      162  STO A      199  GOTO 5'
011  :          069  2          127  GSB d      163  1          200  LBL 7'
012  RCL 5      070  4          128  GSB B      164  .          201  RCL 9
013  +          071  LBL 3'     129  2          165  6          202  GOTO e
014  1          072  -          130  x          166  -          203  LBL 8'
015  2          073  SF 1       131  RTN        167  STO B      204  .
016  x          074  GSB B      132  LBL d      168  RCL A      205  1
017  STO A      075  CF 1       133  2          169  4          206  P≠S
018  9          076  SF 0       134  :          170  .          207  SF 2
019  %          077  GSB d      135  RTN        171  8          208  LBL e
020  STO B      078  STO 9      136  LBL B      172  -          209  4
021  R↑         079  3          137  F? 0       173  X>0        210  ST I
022  ST I       080  x          138  GOTO B'     174  STO B      211  X≠Y
023  R↑         081  GSB C      139  RCL 2      175  4          212  LBL b
024  STO 9      082  RCL 9      140  1          176  ST I       213  RCL B
025  GOTO i     083  5          141  0          177  LBL a      214  x
026  LBL 4      084  x          142  x          178  ISZ        215  RCL i
027  RCL 0      085  X≠Y        143  X≠Y        179  RCL i      216  +
028  ST I       086  STO 9      144  X≤Y        180  RCL A      217  DSZ
029  RCL 8      087  X≠Y        145  SF 2       181  X≤Y        218  GOTO b
030  GSB d      088  GSB C      146  6          182  GOTO i     219  F?  2
031  GOTO 4'     089  RCL 9      147  0          183  RCL 8      220  P≠S
032  LBL 1      090  -          148  F? 2       184  X>Y        221  LBL 5'
033  RCL 0      091  CF 0       149  GSB d      185  GOTO a     222  P≠S
034  LBL c      092  GOTO D     150  :          186  X≠Y        223  INT
035  ST I       093  LBL 7      151  LST X      187  RCL 0      224  RTN
036  RCL 8      094  RCL B      152  X≠Y        188  x
037  LBL 4'     095  RCL C                     189  RCL E
038  STO x 9    096  X>Y
039  RCL 7      097  X≠Y
040  STO + 9    098  RCL 9
041  F? 2       099  X≤Y
042  RTN        100  GOTO 8
043  GSB 7      101  GSB d
044  GSB B      102  X≤Y
045  LBL D      103  GOTO 9
046  1          104  R↓
047  .          105  2
048  2          106  x
049  :          107  LBL E
050  INT        108  SF 1
051  1          109  GSB B
052  0          110  CF 1
053  :          111  RC I
054  R/S        112  +
055  LBL 3      113  RCL A
056  SF 2       114  X≠Y
057  RCL 6      115  -
058  GSB c      116  RTN
```

Bild 3. Programm für den Taschenrechner HP-67 und Belegung der Speicherregister

Belegung der Speicherregister

I : Arbeitsspeicher		R_C	: 4320
R_A : Arbeitsspeicher		R_D	: 1044
R_B : Arbeitsspeicher		R_E	: 13644
R_0	: 1794	R_{S0}	: 5600
R_1	: 15298	R_{S1}	: 2708
R_2	: 4800	R_{S2}	: 2200
R_3	: 109,95	R_{S3}	: 925
R_4	: -6,07	R_{S4}	: -154,42
R_5	: 1,99	R_{S5}	: 0,3719
R_6	: 2544	R_{S6}	: 1,6019
R_7	: 2100	R_{S7}	: 4,7999
R_8	: 600	R_{S8}	: 13,0019
R_9	: Arbeitsspeicher	R_{S9}	: 10,86

Nach Eingabe der Steuerklasse, der Anzahl der Kinder und des Monatseinkommens in das Stack-Register wird das Programm zur Berechnung der Lohnsteuer durch Drücken der Taste „A" gestartet, so daß der monatliche Lohnsteuerbetrag nach einer Rechenzeit von ca. 15 Sekunden angezeigt wird. Ein evtl. vom Finanzamt auf der Steuerkarte eingetragener Steuerfreibetrag wird vom Programm nicht berücksichtigt. Er muß entweder vor dem Starten des Programms vom Monatseinkommen abgezogen oder zur Berücksichtigung durch das Programm als Jahresfreibetrag zu einem der Freibeträge in R_0, R_6 oder R_D (je nach Steuerklasse) addiert werden.

Wegen des großen Programmumfangs konnte die Berechnung der Lohnsteuer nach Steuerklasse 2 nicht mit in das Programm einbezogen werden. Das Programm läßt sich jedoch bei Bedarf so ändern, daß unter Verzicht auf z. B. die Steuerklassen 5 und 6 auch die Steuerklasse 2 programmiert werden kann. Eine einfache Möglichkeit zur Berechnung der Steuerklasse 2 besteht aber auch darin, daß man den Speicherinhalt von R_0 auf 2634 oder auf 4794 ändert (*Tabelle*). Bei Eingabe der Steuerklasse 1 wird jetzt die Steuerklasse 2 berechnet.

Das vollständige Programm und die Belegung der Speicherregister mit den Konstanten und Freibeträgen sind im Listing angegeben.

Schlußbetrachtung

Das beschriebene Programm (*Bild 3*) ermöglicht nach Einlesen zweier Magnetkarten die schnelle Berechnung der Einkommensteuer nach Grund- und Splittingtarif und der monatlichen Lohnsteuer für die Steuerklassen 1 und 3 bis 6 bei beliebiger Anzahl von zu berücksichtigenden Kindern. Nach ausgiebiger Anwendung des Programms haben sich im Vergleich mit Lohnsteuertabellen keine Fehler gezeigt, so daß sich dem Besitzer eines HP-67 eine wertvolle Unterstützung bei der Prüfung der monatlichen Gehaltsabrechnung oder bei der Einkommensteuererklärung bietet.

Das Programm ist so umfangreich, daß alle 224 möglichen Schritte im Programmspeicher ausgenutzt werden mußten. Darüber hinaus mußten auch alle Daten-Speicherregister bis auf die notwendigen Arbeitsspeicher I, R_A, R_B und R_9 mit Konstanten belegt und einige Label mehrfach belegt werden. Die Mehrfachbelegung eines Label ist im Listing zur besseren Übersicht durch Hochkommata (z. B. LBL 1') angedeutet.

HP-41 druckt Kalender

An anderer Stelle in diesem Sonderheft wird beschrieben, wie man mit dem TI-59 einen Jahreskalender ausdrucken kann. Das Problem läßt sich mit dem programmierbaren Taschenrechner HP-41 C nebst zugehörigem Drucker besonders elegant lösen, da dieser über komfortable Möglichkeiten zur alphanumerischen Ein- und Ausgabe verfügt.

Das Programm arbeitet sowohl im Bereich des Julianischen wie auch des Gregorianischen Kalenders absolut korrekt. Vor seiner Eingabe – es erfordert zwei Speichermodule – ist SIZE 020 sowie GTO .. einzutasten. Dann empfiehlt es sich, wie üblich im User-Modus den Eingangslabel „KAL" auf irgendeine Taste zu legen.

Nach dem Start erfolgt die Eingabe der benötigten Daten im Dialogbetrieb bzw. der Taste R/S. „Jahr?" und „Monat?" sind Dialogfragen, deren Beantwortung einen Monatskalender ergibt. Erneut erscheint „Jahr?", und man kann andere Monate wiederholen. Drückt man hierbei R/S ohne Eingabe, so wird beim Monatskalender (Flag 3 im Display) ein Hinweis gedruckt; das Display zeigt ENDE. Mit R/S kann man nun das Programm fortsetzen. Wird bei „Monat?" nichts eingegeben und nur R/S gedrückt, so ergibt sich ein Jahreskalender. Wenn dann auch bei „Jahr?" R/S gedrückt wird, so folgt nur „ENDE" mit Rückstartmöglichkeit durch R/S.

Das Programm benutzt die Steuerflags 0...4 (Display), das Eingabeflag 22 und die Druckerflags 12 und 29. Die Bedeutung der sichtbaren Flags ist:
Flag 0: Monatslängen (außer Februar) gespeichert.
Flag 1: Der Julianische Kalender wird benutzt.
Flag 2: Der Oktober 1582 (Kalenderreform) wird bearbeitet.
Flag 3: Nach Monatskalender; Taste R/S nach „Jahr?" ergibt Hinweis auf HP-41 C.
Flag 4: Druck der ersten Zahlenreihe.

Helmut Moser

```
                PRP "KAL"

 01♦LBL "KAL"
CF 03
 03♦LBL A
CLA  FIX 0  CF 01  CF 04
0  STO 18  "JAHR?"
PROMPT  FS? 22  GTO 00
FS? 03  GTO 26  GTO 24
 17♦LBL 00
STO 15  STO 16  CF 03
CF 22  "MONAT?"  PROMPT
FS? 22  GTO 13  1
 27♦LBL 13
STO 13  STO 14  3  X>Y?
GTO 14  1  ST+ 14
GTO 15
 36♦LBL 14
13  ST+ 14  1  ST- 16
 41♦LBL 15
RCL 16  365.25  *  INT
RCL 14  30.6  *  INT  +
FS? 01  GTO 00  RCL 16
1 E2  /  INT  -  RCL 16
4 E2  /  INT  +
 63♦LBL 00
578172  -  ST+ 18
RCL 18  FS? 01  GTO 00
X>0?  GTO 00  SF 01  -2
STO 18  GTO 15
 76♦LBL 00
7  MOD  X=0?  7  STO 19
28  STO 02  RCL 15  4
MOD  X≠0?  GTO 00  1
ST+ 02  FS? 01  GTO 00
RCL 15  1 E2  MOD  X≠0?
GTO 00  1  ST- 02
RCL 15  4 E2  MOD  X≠0?
GTO 00  1  ST+ 02
107♦LBL 00
FC? 00  XEQ 25  SF 00
CF 29  SF 12  RCL 15
ACX  "  "  ACA  ADV
FS? 01  XEQ 27  ADV
121♦LBL 17
FC? 01  GTO 00  RCL 13
1 E2  /  RCL 15  +
1582.1  X=Y?  SF 02
132♦LBL 00
1  STO 00  SF 04  SF 12
XEQ IND 13  ACA  ADV
CF 12  "MO DI MI DO"
ACA  " FR SA SO"  ACA
ADV  8  RCL 19  -  X=0?
7  1 E-3  *  1  +
STO 17
156♦LBL 18
FS? 04  GTO 19  "  "
ACA  1.007  STO 17
GTO 20
164♦LBL 19
FC? 02  GTO 20  RCL 00
5  X≠Y?  GTO 20  10
ST+ 00  CF 02
174♦LBL 20
9  RCL 00  X>Y?  GTO 00
"  "  ACA
181♦LBL 00
RCL 00  ACX  RCL IND 13
X=Y?  GTO 22  1  ST+ 00
ISG 17  GTO 19  FS?C 04
GTO 21  PRBUF  GTO 18
195♦LBL 21
ADV  GTO 18
198♦LBL 22
PRBUF  FS? 22  GTO 23  1
ST+ 13  ST+ 17  13
RCL 13  X=Y?  GTO 23
RCL 17  INT  STO 19
CF 00  PRBUF  GTO 17
215♦LBL 23
ADV  FS?C 22  SF 03
FS? 03  GTO A  GTO 26
222♦LBL 24
ADV  ADV  CLX  BEEP
BEEP  "* * ENDE * *"
PROMPT  GTO "KAL"
231♦LBL 25
30  STO 04  STO 06
STO 09  STO 11  31
STO 01  STO 03  STO 05
STO 07  STO 08  STO 10
STO 12  RTN
246♦LBL 26
"DIESER KALENDER"  ACA
" WURDE"  ACA  ADV
"MIT EINEM"  ACA  SF 12
" HP-41C"  ACA  ADV
CF 12  "BERECHNET UND"
ACA  " GEDRUCKT."  ACA
ADV  FC?C 03  GTO "KAL"
GTO 24
267♦LBL 27
ADV  CF 12
"DER VERLANGTE "  ACA
"KALENDER"  ACA  ADV
"LIEGT ZEIT"  ACA
"LICH VOR DER"  ACA  ADV
"GREGOR. KALEN"  ACA
"DERREFORM"  ACA  ADV
"    IM "  ACA  XEQ 10
ACA  1582  ACX  "."  ACA
PRBUF  RTN
295♦LBL 01
"JANUAR  "  RTN
298♦LBL 02
"FEBRUAR "  RTN
301♦LBL 03
"MAERZ  "  RTN
304♦LBL 04
"APRIL  "  RTN
307♦LBL 05
"MAI    "  RTN
310♦LBL 06
"JUNI   "  RTN
313♦LBL 07
"JULI   "  RTN
316♦LBL 08
"AUGUST  "  RTN
319♦LBL 09
"SEPTEMBER"  RTN
322♦LBL 10
"OKTOBER "  RTN
325♦LBL 11
"NOVEMBER "  RTN
328♦LBL 12
"DEZEMBER "  RTN  END
```

Programm zum Ausdruck eines Kalenders nach dem Julianischen oder Gregorianischen Prinzip mit dem programmierbaren Taschenrechner HP-41

Kalenderausdruck mit TI-59

In Heft 22/1979 der FUNKSCHAU fand sich auf Seite 1312 ein Programm, das es gestattet, mit der Kombination TI-59/PC-100 einen Jahreskalender auszudrucken. Im folgenden wird eine um 40 % verkürzte und gleichzeitig um 60 % schnellere Version beschrieben. Ferner ist es auch möglich, mehrmals hintereinander den gleichen Ausdruck zu liefern.

Bild 1 zeigt das vollständige Programm und die Inhalte der Datenregister. Da die Speicherinhalte zum Teil mehr als zehn Stellen haben, müssen sie in zwei Stufen eingegeben werden: Zunächst muß man den ganzzahligen Teil mit dem Nachkommateil addieren und dann erst in das jeweilige Register abspeichern.

Die Eingabe des Startdatums erfolgt in der Form MM,JJJJ; zum Beispiel 11, 1979 für November 1979. Wenn der Kalender mit dem Monat Januar beginnen soll, braucht man nur z.B. ",1980" eingeben. Gestartet wird das Programm mit der Label-Taste A.

Soll der Kalender mehrfach ausgedruckt werden, so tippt man die gewünschte Anzahl ein und drückt dann einfach B. Mit der Taste C kann man noch eingeben, wieviele Jahre nacheinander ausgedruckt werden sollen. Werden B und C nicht betätigt oder wird dabei Null eingegeben, so nimmt der Rechner für beide Werte eins an.

```
                1980.

      J A N U A R

   MO DI MI DO FR SA SO

      01 02 03 04 05 06
   07 08 09 10 11 12 13
   14 15 16 17 18 19 20
   21 22 23 24 25 26 27
   28 29 30 31
```

Bild 2. Teilausdruck für den Januar 1980

```
000  76 LBL     046  44 SUM     092  52 EE      138  02  02     184  59 INT     230  43 RCL
001  18 C'      047  00  00     093  69 OP      139  95  =      185  77 GE      231  07  07
002  04  4      048  73 RC*     094  02  02     140  52 EE      186  01  01     232  65  ×
003  42 STO     049  02  02     095  85  +      141  01  1      187  89  89     233  05  5
004  02  02     050  22 INV     096  92 RTN     142  00  0      188  01  1      234  00  0
005  73 RC*     051  59 INT     097  52 EE      143  22 INV     189  42 STO     235  33 X²
006  03  03     052  69 OP      098  01  1      144  52 EE      190  00  00     236  95  =
007  69 OP      053  22  22     099  22 INV     145  69 OP      191  52 EE      237  22 INV
008  23  23     054  71 SBR     100  52 EE      146  03  03     192  02  2      238  59 INT
009  84 OP*     055  40 IND     101  75  -      147  85  +      193  22 INV     239  69 OP
010  02  02     056  00  00     102  61 GTO     148  92 RTN     194  52 EE      240  10  10
011  97 DSZ     057  97 DSZ     103  01  01     149  76 LBL     195  85  +      241  94 +/-
012  02  02     058  01  01     104  37  37     150  17 B'      196  32 X:T     242  75  -
013  00  00     059  00  00     105  52 EE      151  95  =      197  52 EE      243  10 E'
014  05  05     060  45  45     106  05  5      152  42 STO     198  04  4      244  10 E'
015  69 OP      061  32 X:T     107  22 INV     153  01  01     199  22 INV     245  19 D'
016  05  05     062  06  6      108  52 EE      154  73 RC*     200  52 EE      246  10 E'
017  98 ADV     063  05  5      109  95  =      155  01  01     201  99 PRT     247  19 D'
018  92 RTN     064  67  EQ     110  69 OP      156  22 INV     202  98 ADV     248  10 E'
019  76 LBL     065  00  00     111  03  03     157  59 INT     203  01  1      249  10 E'
020  10 E'      066  69  69     112  92 RTN     158  52 EE      204  95  =      250  19 D'
021  01  1      067  69 OP      113  52 EE      159  03  3      205  36 PGM     251  10 E'
022  76 LBL     068  05  05     114  09  9      160  22 INV     206  20  20     252  19 D'
023  19 D'      069  98 ADV     115  22 INV     161  52 EE      207  14  D      253  10 E'
024  85  +      070  98 ADV     116  52 EE      162  92 RTN     208  85  +      254  98 ADV
025  03  3      071  25 CLR     117  69 OP      163  76 LBL     209  05  5      255  97 DSZ
026  00  0      072  92 RTN     118  04  04     164  13  C      210  01  1      256  04  04
027  95  =      073  52 EE      119  85  +      165  42 STO     211  17 B'      257  01  01
028  42 STO     074  09  9      120  92 RTN     166  06  06     212  48 EXC     258  76  76
029  01  01     075  22 INV     121  52 EE      167  91 R/S     213  00  00     259  04  4
030  18 C'      076  52 EE      122  03  3      168  76 LBL     214  85  +      260  22 INV
031  05  5      077  69 OP      123  22 INV     169  11  A      215  42 STO     261  28 LOG
032  06  6      078  01  01     124  52 EE      170  42 STO     216  03  03     262  35 1/X
033  48 EXC     079  85  +      125  95  =      171  07  07     217  04  4      263  44 SUM
034  03  03     080  92 RTN     126  69 OP      172  43 RCL     218  49 PRD     264  07  07
035  32 X:T     081  52 EE      127  04  04     173  05  05     219  03  03     265  97 DSZ
036  18 C'      082  03  3      128  69 OP      174  42 STO     220  44 SUM     266  06  06
037  32 X:T     083  22 INV     129  05  05     175  04  04     221  03  03     267  01  01
038  42 STO     084  52 EE      130  69 OP      176  43 RCL     222  03  3      268  72  72
039  03  03     085  95  =      131  00  00     177  07  07     223  08  8      269  76 LBL
040  69 OP      086  69 OP      132  06  6      178  22 INV     224  17 B'      270  12  B
041  00  00     087  01  01     133  05  5      179  59 INT     225  42 STO     271  42 STO
042  08  8      088  92 RTN     134  42 STO     180  85  +      226  01  01     272  05  05
043  42 STO     089  52 EE      135  00  00     181  32 X:T     227  83 GO*     273  91 R/S
044  02  02     090  07  7      136  92 RTN     182  43 RCL     228  01  01
045  08  8      091  22 INV     137  69 OP      183  07  07     229  10 E'
```

Bild 1. Vollständiges Programmlisting und Registerinhalte. Bei der Eingabe der Speicherwerte ist zu beachten, daß hier auch die Rechnerschutzstellen verwendet werden, so daß sie erst nach Addition von Vor- und Nachkommateil möglich ist

```
SPEICHERINHALTE
          .102    08
3500000000.103    09
  41001300.104    10
2500130031.105    11
          .106    12
1300350000.107    13
  35004100.11     14
2100170014.111    15
          .112    16
          .201    17
  35004600.202    18
3000130017.203    19
          .204    20
          .205    21
  24002700.206    22
1300330035.207    23
          .21     24
          .211    25
          .212    26
3000130024.301    27
          .302    28
          .303    29
  24000000.304    30
2500410031.305    31
          .306    32
          .307    33
  24000000.31     34
2500410027.311    35
          .312    36
3700000000.401    37
  41003600.402    38
1300410022.229    39
  35000000.23     40
3000140017.244    41
  37001700.245    42
3600170033.246    43
          .247    44
1700350000.248    45
  32001400.249    46
3200260037.25     47
          .251    48
1400170035.252    49
  17003000.253    50
3100320042.105    51
          .113    52
1400170035.065    53
  17003000.073    54
1600170046.081    55
3613003632.089    56
3200213500.097    57
  30240016.       58
3032001624.       59
```

Noch eine Bemerkung zu den Änderungen gegenüber der ursprünglichen FUNKSCHAU-Version. Es wird manchmal behauptet, daß sich die Rechengeschwindigkeit erhöht, wenn man Labels durch absolute Adressierung ersetzt. Dies ist nur eingeschränkt richtig. Befinden sich die Labels nämlich in den Programmschritten 000...035, so ist die Labeladressierung schneller.

Klaus Neumann

PET, der Unergründete

Wenig bekannte Programmiertricks

Zwar kennt der PET-2001 die meisten Basic-Befehle, die bei Kleincomputern seiner Größe gebräuchlich sind; um aber alle seine Möglichkeiten ausnutzen zu können, sollte man auch die hier beschriebenen Tricks kennen, die speziell auf den PET zutreffen, aber nicht in allen Fällen auf Rechner anderer Hersteller übertragen werden können. Vielleicht wird deswegen darüber in PET-Betriebshandbüchern so wenig geschrieben.

So kann man in Commodore-Basic logische Vergleiche auch ohne IF...THEN... ausführen. Der PET definiert nämlich bei logischen Vergleichen das „Falsch" einer Aussage mit 0, und „Wahr" mit –1. Der Wert –1 für „Wahr" wurde gewählt, da NOT (–1) = 0 und NOT (0) = –1 ist, und sich somit das Ergebnis eines Vergleichs leicht mit dem logischen Operator NOT negieren läßt. Diese Wertänderung zwischen 0 und –1 ist bedingt durch die bitweise Komplementierung, die auch das Vorzeichenbit betrifft.

So ergibt der Ausdruck PRINT(A=5) für A=5 den „Wahr"-Wert –1, für A=5 den „Falsch"-Wert 0. Auf diese Weise ließen sich folgende Basic-Programme auf die unten ausgeführte Weise verbessern:

Altes Programm mit
IF...THEN...
.
.
.
100 IF A <=4THEN B=–25 : GOTO 120
110 B=–15
120 PRINT B
.
.
.

Neues Programm ohne
IF...THEN...
.
.
.
100 B= (A < = 4) x 10 –15 : PRINT B
.
.
.
.
.
.
.

Die Variable B erhält auf beide Arten den Wert –25 für A < = 4 und –15 für A > 4. Daraus wird auch deutlich, warum der PET keine Doppelwertzuweisung wie B=A=10 versteht, denn sein Betriebssystem erkennt darin B=(A=10), und weist der Variablen B entweder den Wert 0 oder –1 zu. Auch bei Textverarbeitung erweist sich die Programmierung mit diesem Befehl als nützlich, z. B.:

Altes Programm
10 X$(1) = „KATZE": X$(2) = „HUND"
.
.
.
50 IF A$ = LEFT$ (B$, 7) THEN
PRINTX$ (1) : GOTO70
60 PRINT X$(2)
70...
.
.
.

Neues Programm
10 X$(1) = „KATZE": X$(2) = „HUND"
.
.
.
50 PRINTX$((A$=LEFT$ (B$, 7))+2)
.
.
.

Wie bei IF...THEN... sind auch bei diesem Vergleichsbefehl alle logischen Operatoren programmierbar. Im folgenden Beispiel wird dies in einer logischen Entscheidung für einen bedingten Sprung mit zwei Sprungadressen benutzt.

.
.
.
500 IF C>11 AND B<=–4GOTO60
510 GOTO80
.
.
.
500 ON (C>11AND B<=–4)+2
GOTO 60, 80

Auch die folgende Anweisung ist nützlich und bisher nicht beschrieben worden:

...IF A THEN... entspricht
...IF A <> 0 THEN...

...IF A$ THEN... entspricht
...IF A$ <>" " THEN...

bzw. ...IF LEN(A$) <> 0 THEN...

Dabei bedeutet " " den Nullstring. Diese verkürzte Art der IF...THEN-Anweisung kann auch in folgender Form verwendet werden:

IF X AND C+1 THEN... entspricht... IFX <> 0 AND C<> –1THEN...

Zum Schluß soll noch auf die Möglichkeit des GET-Befehls mit Cursor hingewiesen werden. Der Cursor des PET-2001 läßt sich mit POKE 548,0 bzw. mit POKE 548,1 auch während des Programmablaufs ein- bzw. ausschalten. Für ein einmaliges GET ergibt sich somit z. B.: POKE 548,0 : POKE 525,0 : WAIT 525,1 : GETA$: POKE 548,1.

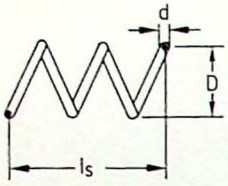
Frank Fleischmann

TI-59 berechnet Spulen

Bereits in Heft 7/1979 der FUNKSCHAU auf Seite 400 veröffentlichte Fred H. Heller ein TI-59-Programm für die Berechnung einlagiger Zylinderspulen. Das in *Bild 1* aufgelistete Programm ist aus diesem abgeleitet und stellt eine verbesserte Version dar. Die Bedienung geht aus *Bild 2* hervor, während *Bild 3* darin verwendete Formeln enthält, mit Hilfe derer es keine Schwierigkeit sein dürfte, entsprechende Programme für HP-Rechner

000	76	LBL	069	01	1	138	05	5	207	95	=	277	05	5	347	01	1
001	16	A'	070	00	0	139	65	×	208	34	ΓX	278	65	×	348	95	=
002	06	6	071	00	0	140	00	0	209	42	STO	279	43	RCL	349	42	STO
003	09	9	072	00	0	141	93	.	210	07	07	280	06	06	350	14	14
004	93	.	073	00	0	142	05	5	211	91	R/S	281	75	-	351	87	IFF
005	05	5	074	00	0	143	65	×	212	76	LBL	282	89	π	352	01	01
006	42	STO	075	00	0	144	43	RCL	213	13	C	283	33	X²	353	03	03
007	04	04	076	95	=	145	08	08	214	67	EQ	284	65	×	354	63	63
008	06	6	077	42	STO	146	95	=	215	02	02	285	43	RCL	355	91	R/S
009	02	2	078	09	09	147	42	STO	216	20	20	286	06	06	356	76	LBL
010	93	.	079	55	÷	148	11	11	217	42	STO	287	33	X²	357	10	E'
011	05	5	080	01	1	149	33	X²	218	08	08	288	65	×	358	86	STF
012	42	STO	081	00	0	150	85	+	219	91	R/S	289	43	RCL	359	01	01
013	00	00	082	00	0	151	43	RCL	220	89	π	290	07	07	360	61	GTO
014	91	R/S	083	00	0	152	08	08	221	33	X²	291	33	X²	361	03	03
015	42	STO	084	00	0	153	65	×	222	65	×	292	55	÷	362	24	24
016	15	15	085	00	0	154	43	RCL	223	43	RCL	293	43	RCL	363	55	÷
017	91	R/S	086	00	0	155	10	10	224	06	06	294	08	08	364	01	1
018	76	LBL	087	95	=	156	55	÷	225	33	X²	295	54	)	365	00	0
019	18	C'	088	34	ΓX	157	43	RCL	226	65	×	296	95	=	366	00	0
020	06	6	089	35	1/X	158	12	12	227	43	RCL	297	42	STO	367	55	÷
021	07	7	090	65	×	159	95	=	228	07	07	298	13	13	368	43	RCL
022	42	STO	091	43	RCL	160	34	ΓX	229	33	X²	299	33	X²	369	05	05
023	04	04	092	04	04	161	85	+	230	55	÷	300	85	+	370	55	÷
024	05	5	093	95	=	162	43	RCL	231	53	(	301	00	0	371	43	RCL
025	08	8	094	55	÷	163	11	11	232	43	RCL	302	93	.	372	15	15
026	42	STO	095	01	1	164	95	=	233	10	10	303	00	0	373	55	÷
027	00	00	096	00	0	165	42	STO	234	85	+	304	00	0	374	89	π
028	91	R/S	097	00	0	166	06	06	235	00	0	305	03	3	375	55	÷
029	42	STO	098	00	0	167	91	R/S	236	93	.	306	65	×	376	43	RCL
030	15	15	099	95	=	168	76	LBL	237	04	4	307	43	RCL	377	00	00
031	91	R/S	100	42	STO	169	11	A	238	05	5	308	06	06	378	95	=
032	76	LBL	101	05	05	170	67	EQ	239	65	×	309	33	X²	379	35	1/X
033	17	B'	102	91	R/S	171	01	01	240	43	RCL	310	95	=	380	65	×
034	29	CP	103	76	LBL	172	76	76	241	06	06	311	34	ΓX	381	02	2
035	43	RCL	104	14	D	173	42	STO	242	75	-	312	85	+	382	65	×
036	01	01	105	67	EQ	174	07	07	243	00	0	313	43	RCL	383	89	π
037	67	EQ	106	01	01	175	91	R/S	244	93	.	314	13	13	384	65	×
038	00	00	107	11	11	176	43	RCL	245	00	0	315	95	=	385	43	RCL
039	57	57	108	42	STO	177	08	08	246	00	0	316	42	STO	386	09	09
040	53	(	109	06	06	178	55	÷	247	03	3	317	10	10	387	65	×
041	43	RCL	110	91	R/S	179	61	GTO	248	65	×	318	91	R/S	388	43	RCL
042	03	03	111	53	(	180	33	X²	249	43	RCL	319	76	LBL	389	08	08
043	65	×	112	89	π	181	65	×	250	06	06	320	19	D'	390	65	×
044	43	RCL	113	33	X²	182	53	(	251	33	X²	321	22	INV	391	00	0
045	01	01	114	65	×	183	43	RCL	252	55	÷	322	86	STF	392	93	.
046	95	=	115	43	RCL	184	10	10	253	43	RCL	323	01	01	393	00	0
047	35	1/X	116	07	07	185	85	+	254	10	10	324	43	RCL	394	00	0
048	85	+	117	33	X²	186	43	RCL	255	54	)	325	07	07	395	00	0
049	43	RCL	118	85	+	187	06	06	256	95	=	326	75	-	396	00	0
050	02	02	119	00	0	188	65	×	257	42	STO	327	01	1	397	00	0
051	55	÷	120	93	.	189	00	0	258	08	08	328	95	=	398	00	0
052	43	RCL	121	00	0	190	93	.	259	91	R/S	329	35	1/X	399	00	0
053	01	01	122	00	0	191	04	4	260	76	LBL	330	65	×	400	00	0
054	54	)	123	03	3	192	05	5	261	12	B	331	43	RCL	401	01	1
055	35	1/X	124	65	×	193	75	-	262	67	EQ	332	10	10	402	95	=
056	95	=	125	43	RCL	194	00	0	263	02	02	333	95	=	403	42	STO
057	91	R/S	126	08	08	195	93	.	264	68	68	334	33	X²	404	03	03
058	42	STO	127	55	÷	196	00	0	265	42	STO	335	85	+	405	91	R/S
059	01	01	128	43	RCL	197	00	0	266	10	10	336	43	RCL	406	76	LBL
060	91	R/S	129	10	10	198	03	3	267	91	R/S	337	06	06	407	33	X²
061	42	STO	130	54	)	199	65	×	268	00	0	338	33	X²	408	43	RCL
062	02	02	131	42	STO	200	43	RCL	269	93	.	339	95	=	409	06	06
063	61	GTO	132	12	12	201	06	06	270	05	5	340	34	ΓX	410	33	X²
064	00	00	133	35	1/X	202	33	X²	271	94	+/-	341	65	×	411	55	÷
065	35	35	134	65	×	203	55	÷	272	65	×	342	89	π	412	89	π
066	76	LBL	135	00	0	204	43	RCL	273	53	(	343	65	×	413	33	X²
067	15	E	136	93	.	205	10	10	274	00	0	344	43	RCL	414	61	GTO
068	65	×	137	04	4	206	54	)	275	93	.	345	07	07	415	01	01
									276	04	4	346	85	+	416	81	81

Bild 1. Das vollständige TI-59-Programm zur Spulenberechnung. Es hat auf zwei Magnetkartenseiten, d. h. auf einer Karte Platz



Eingabe	Taste	Anzeige
N [Windungen]	A	N
l_s [cm] Spulenlänge	B	l_s
L [nHy] Spuleninduktivität	C	L
D [cm] Spulendurchmesser	D	D
——	D'	l_D

$$N = \sqrt{\frac{L}{D^2 \cdot \pi^2}\left(\ell + 0.45 D - 0.003 \frac{D^2}{\ell}\right)} \qquad l_D = N \cdot \pi \cdot \sqrt{D^2 + \left(\frac{l_s}{2N}\right)^2} + 1\,[\text{cm}] \qquad \varkappa = \frac{1}{\alpha}$$

α = spez. Widerstand des Spulendrahtes

$$D = \frac{0.45 \cdot 0.5 \cdot L}{\pi^2 \cdot N^2 + 0.003 \cdot \frac{L}{\ell}} + \sqrt{\left(\frac{0.45 \cdot 0.5 \cdot L}{\pi^2 \cdot N^2 + 0.003 \frac{L}{\ell}}\right)^2 + \frac{L \cdot \ell}{\pi^2 \cdot N^2 + 0.003 \frac{L}{\ell}}}\,[\text{cm}]$$

Bild 3. Verwendete Formeln für die Berechnung der Windungszahl, des mittleren Spulendurchmessers und der spezifischen Leitfähigkeit des Drahtes (α ist der spezifische Widerstand)

Bei Silber- bzw. versilbertem Draht

Eingabe	Taste	Anzeige
—	A'	62,5
d [mm] Draht Ø	R/S	d
f [MHz]	E	$\frac{695}{\sqrt{f}}$
—	E'	Q_{LL}
—	B'	0
$\mu' = \mu_{rel}$ des Kerns	R/S	0
μ''-Verlustbeiwert d. Kerns	R/S	Q_{LK}

Bei Kupferdraht

Eingabe	Taste	Anzeige
—	C'	58
d [mm] Draht Ø	R/S	d
f [MHz]	E	$\frac{67}{\sqrt{f}}$
—	E'	Q_{LL}
—	B'	0
$\mu' = \mu_{rel}$ des Kerns	R/S	μ'
μ''-Verlustbeiwert d. Kerns	R/S	Q_{LK}

◀ Bild 2. Schema zur Eingabe der bekannten Größen. Sind sie alle eingetippt, so setzt man die gesuchte Größe Null, und schon wird das Ergebnis errechnet

oder in Basic zu schreiben. Für die jeweils gesuchte Größe X (N, l_s, L, D) ist Null einzugeben. Ein Beispiel: Gesucht sei L, d.h. die Induktivität einer Spule mit bekannten Abmessungen. Dafür gibt man zunächst alle bekannten Größen ein, dann drückt man Null und die der Induktivität zugeordnete Taste C.

Dipl.-Ing. Ernst Sohn

Hex-Monitor für TRS-80

Das in *Bild 1* abgedruckte BASIC-Programm ist die TRS-80-Version des in FUNKSCHAU Heft 2/80 beschriebenen Hex-Monitors. Es läuft auf Level-II-Maschinen und – mit den angegebenen Änderungen – auch in Disk-BASIC.

Als Bedienungshilfe kann man zusätzlich die Betriebsart „HEX" einfügen, in der man vierstellige Hexadezimalzahlen in Dezimalzahlen umrechnet (*Bild 2*). Das ist besonders nützlich im Bereich oberhalb 7FFF, da die Argumente für den POKE-Befehl hier negativ werden. Man verläß diese Betriebsart, indem man X drückt.

In der Betriebsart „POKE" wird neben der Adresse der augenblickliche Inhalt der Speicherzelle angezeigt. Will man ihn unverändert lassen, drückt man „Enter", und die nächste Adresse wird mit Inhalt angezeigt. Um einen neuen Wert einzuschreiben, tippt man ihn wie im Originalprogramm ein.

Rudolf Hofer

```
ZUSATZ FUER HEX/DEZ-WANDLUNG
65 IF A$="HEX"THEN7oo
7oo INPUT "HEX";K$:IF K$="X" GOTO6o
7o5 S=o
71o FOR I=o TO 3
72o FOR J=o TO 15:IF MID$(K$,I+1,1)=H$(J) THEN74o
73o NEXT J
74o S=S+Jx16↑(3-I)
75o NEXT I
76o IF S>32767 THEN S=S-65536
765 PRINT INT (S+o.7)
77o GOTO7oo
```

Bild 2. Hilfsprogramm zum Umrechnen vierstelliger Hex-Zahlen in Dezimalzahlen

Literatur

[1] *Feichtinger, H.*: Ein Hex-Monitor in BASIC. FUNKSCHAU 1980, Heft 2, S. 83.

```
 1o REM xxxHEX-MONITORxxx
 2o REM COMMANDS: POKE,LIST,RUN
 3o DIM H$(16):T=1oo
 4o DATA o,1,2,3,4,5,6,7,8,9,A,B,C,D,E,F
 5o FOR I=o TO 15:READ H$(I):NEXT
 6o INPUT "CMD=";A$
 7o INPUT "DEC.ADR.: FROM=";B
 8o IF A$="LIST"GOTO 2oo
 9o IF A$="POKE"GOTO 4oo
1oo IF A$="RUN"GOTO 6oo
11o GOTO 6o
2oo INPUT "TO=";C
21o FOR I=B TO C:D=PEEK(I)
22o E=INT(D/16): F=D AND 15
23o PRINT I;H$(E)+H$(F)
24o FOR J=o TO T:D$=INKEY$
25o IF D$="Q" THEN T=1o
26o IF D$="S" THEN T=1oo
27o IF D$="X" GOTO 6o
28o NEXT J
29o NEXT I
3oo GOTO 6o
4oo PRINT"    ":PRINT B;
4o2 D=PEEK(B):E=INT(D/16):F=D AND 15
4o4 PRINT H$(E)+H$(F);" ";
4o6 GOSUB 5oo
4o8 IF D$=CHR$(13) GOTO43o
41o D=16xI:GOSUB5oo
42o D=D+I:POKE B,D
43o B=B+1:GOTO4oo
5oo D$=INKEY$:IF D$="X"THEN CLS:GOTO6o
5o5 IF D$=CHR$(13) RETURN
51o IF D$k"o"OR D$g"F"GOTO5oo
52o FOR I=o TO 15:IF D$=H$(I)GOTO54o
53o NEXT
54o PRINT D$;:RETURN
598 REM TRS-8o-USR-ADR.16526/27 LSB/MSB
6oo POKE 16526,B AND 255:POKE 16527, INT(B/256)
62o PRINT "RESULT:";:PRINT USR(o):GOTO6o

AENDERUNG FUER DISK-BASIC

6oo DEFUSR1=B
61o INPUT "ARGUMENT";A
62o PRINT "RESULT:";:PRINT USR1(A):GOTO6o
```

Bild 1. TRS-80-Version des in FUNKSCHAU Heft 2/1980 abgedruckten Hex-Monitors in Basic. Der Ausdruck entstand mit einer Schreibmaschine, wobei einige ASCII-Zeichen anders interpreitert werden: §=$, o=0, x=*, k=<, g=>

Das nachfolgende Taschenrechnerprogramm erlaubt die überschlägige Dimensionierung der Schaltelemente eines einstufigen Verstärkers in Emitterschaltung mit kapazitiv angekoppelter Last. Dabei wird auch der günstigste Kompromiß zwischen Aussteuerbarkeit und Stromverbrauch errechnet.

Norbert Ulamec

TI-59 dimensioniert Verstärker

Ein Verstärker ist für symmetrische Signale dann optimal dimensioniert, wenn er bei geringstmöglichem Stromverbrauch die geforderte Ausgangsleistung am Lastwiderstand gerade abgeben kann, ohne daß Signalverzerrungen auftreten. Bei Übersteuerungen des Verstärkers soll die Signalbeschneidung symmetrisch sein, d. h. gleichmäßig an den oberen und unteren Spitzen. Diese Betriebsart ist der sogenannte „A-Betrieb". Für die Dimensionierung wurde die Emitterschaltung wegen ihrer guten Leistungsverstärkung gewählt, *Bild 1* zeigt die Standardsschaltung.

Das Programm *(Bild 2)* kann auf dem Rechner TI-59 angewendet werden. Für den Verstärker können alle Bauteile mit Ausnahme des Transistors, der aus einem Datenbuch ermittelt wird, berechnet werden.

Die errechneten Werte werden bereits in der E-12-Normreihe ausgegeben und bei Verwendung eines Druckers (PC 100 B/A) auf dem Ausdruck festgehalten. Die Kondensatoren werden nicht genormt, da man den tatsächlichen Wert deutlich größer als den errechneten Wert wählen soll.

Vor dem Eingeben des Programms ist es wichtig, die Speicherbereichsverteilung auf 559/49 zu ändern. Die zur Aufzeichnung erforderlichen Magnetkarten können ohne Änderung der Bereichsverteilung nicht wieder eingelesen werden.

Da der Rechner unter dem Programmablauf eine Datenkarte mit den Normwerten anfordert, ist es zweckmäßig, die Werte von Widerständen und Batteriespannungen *(Tabelle)* auf einer Magnetkarte unter Block 4 zu speichern.

Tabelle: Inhalt der Datenregister

Nr.	Wert	Nr.	Wert	Nr.	Wert
0	0	10	4,7	20	18
1	0	11	5,6	21	24
2	1	12	6,8	22	32
3	1,2	13	8,2	23	48
4	1,5	14	10	24	100
5	1,8	15	5		
6	2,2	16	6		
7	2,7	17	8		
8	3,3	18	12		
9	3,9	19	15		

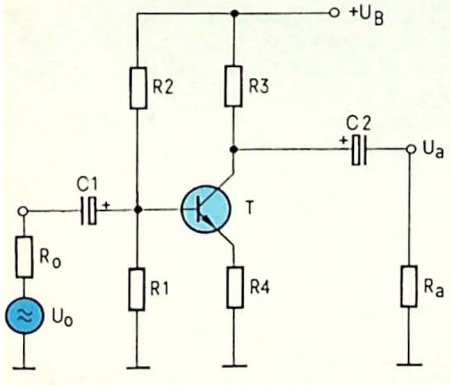


Bild 1. Standardschaltung für einen einfachen Verstärker

```
PROGRAMMAUFLISTUNG
EMITTERSCHALTUNG

000  76 LBL
001  11  A
002  69 OP
003  00  00
004  01  1
005  07  7
006  04  4
007  00  0
008  00  0
009  00  0
010  03  3
011  06  6
012  04  4
013  00  0
014  69 OP
015  01  01
016  69 OP
017  05  05
018  98 ADV
019  04  4
020  22 INV
021  96 WRT
022  00  0
023  91 R/S
024  76 LBL
025  12  B
026  48 EXC
027  25  25
028  48 EXC
029  29  29
030  48 EXC
031  28  28
032  48 EXC
033  27  27
034  48 EXC
035  26  26
036  91 R/S
037  76 LBL
038  13  C
039  00  0
040  32 X:T
041  43 RCL
042  10  10
043  67  EQ
044  54  )
045  25 CLR
046  04  4
047  01  1
048  01  1
049  04  4
050  69 OP
051  04  04
052  25 CLR
053  01  1
054  05  5
055  42 STO
056  00  00
057  43 RCL
058  27  27
059  55  ÷
060  43 RCL
061  26  26
062  95  =
063  42 STO
064  30  30
065  02  2
066  93  .
067  05  5
068  85  +
069  02  2
070  55  ÷
071  43 RCL
072  30  30
073  95  =
074  65  ×
075  43 RCL
076  27  27
077  85  +
078  43 RCL
079  29  29
080  95  =
081  32 X:T
082  76 LBL
083  22 INV
084  73 RC*
085  00  00
086  77  GE
087  23 LNX
088  69 OP
089  20  20
090  61 GTO
091  22 INV
092  76 LBL
093  23 LNX
094  42 STO
095  31  31
096  69 OP
097  06  06
098  91 R/S
099  76 LBL
100  14  D
101  25 CLR
102  03  3
103  05  5
104  00  0
105  04  4
106  69 OP
107  04  04
108  71 SBR
109  45 YX
110  43 RCL
111  31  31
112  75  -
113  43 RCL
114  29  29
115  95  =
116  55  ÷
117  43 RCL
118  27  27
119  75  -
120  02  2
121  75  -
122  02  2
123  55  ÷
124  43 RCL
125  30  30
126  95  =
127  65  ×
128  43 RCL
129  28  28
130  95  =
131  71 SBR
132  24 CE
133  32 X:T
134  76 LBL
135  32 X:T
136  73 RC*
137  01  01
138  22 INV
139  77  GE
140  33 X²
141  69 OP
142  31  31
143  61 GTO
144  32 X:T
145  76 LBL
146  33 X²
147  65  ×
148  43 RCL
149  34  34
150  95  =
151  42 STO
152  32  32
153  69 OP
154  06  06
155  91 R/S
156  76 LBL
157  15  E
158  25 CLR
159  03  3
160  05  5
161  00  0
162  05  5
```

Bild 2. Auflistung des TI-59-Programms. Die Speicherverteilung ist auf 559/49 zu ändern ▶

Die Bedienung geht wie folgt vor sich: Die Taste A am Rechner wird gedrückt; ist der Rechner auf den Drucker gesetzt, so wird die Überschrift gedruckt. Als nächstes fordert der Rechner die Datenkarte an. Nach der Eingabe erscheint 0 in der Anzeige, und die manuelle Dateneingabe auf Label B kann erfolgen. Die Daten von Eingangsspannung U_e, Ausgangsspannung U_a, Lastwiderstand R_a, Sättigungsspannung U_{sat} und Basis-Emitter-Spannung U_{BE} werden nacheinander auf B eingegeben. Die Batteriespannung U_B errechnet sich auf Label C, der Widerstand R 3 auf Label D und der Widerstand R 4 auf Label E. Auf Label A' wird zuerst der Kollektorstrom I_C errechnet; drückt man nun mehrmals R/S, so erscheinen die Werte von Kollektor-Emitter-Spannung U_{CE}, Leistung P_C des Transistors und die Basisspannung gegen Masse U_{BO}. Jetzt muß eine neue Dateneingabe, nämlich jene des ausgewählten Transistors, erfolgen. Auf Label B' werden wiederum nacheinander der Faktor α, der h-Parameter h_{21e} und die untere Grenzfrequenz f_u eingegeben, gefolgt von R/S. Auf Label C' wird nun R 1 errechnet und mit R/S anschließend R 2 ermittelt. Der Kondensator C 1 errechnet sich auf Label D' und mit R/S C 2. Will man noch die maximal zu übertragende Frequenz errechnen, so gibt man den Wert für die Schaltkapazität C_a des Transistors ein und drückt E'. Das Programm ist kontinuierlich aufgebaut, so daß die Werte in der angegebenen Reihenfolge errechnet werden müssen. *Bild 3* zeigt ein Beispiel einer typischen Vorverstärker-Stufe.

```
163  69 DP     244  65  ×      325  69 DP     406  95  =      487  42 STO
164  04  04    245  43 RCL     326  06  06    407  71 SBR     488  47  47
165  01  1     246  37  37     327  91 R/S     408  24 CE      489  02  2
166  00  0     247  95  =      328  76 LBL     409  32 X:T     490  01  1
167  42 STO    248  94 +/-     329  18 C'      410  76 LBL     491  69 DP
168  34  34    249  85  +      330  25 CLR     411  52 EE      492  04  04
169  02  2     250  43 RCL     331  03  3     412  73 RC*     493  43 RCL
170  42 STO    251  31  31     332  05  5     413  01  01    494  28  28
171  01  01    252  95  =      333  00  0     414  22 INV     495  35 1/X
172  43 RCL    253  42 STO     334  02  2     415  77  GE     496  85  +
173  32  32    254  38  38     335  69 DP     416  53  (      497  43 RCL
174  35 1/X    255  69 DP      336  04  04    417  69 DP     498  32  32
175  85  +     256  06  06     337  71 SBR     418  31  31    499  35 1/X
176  43 RCL    257  91 R/S     338  45 YX      419  61 GTO     500  95  =
177  28  28    258  03  3      339  43 RCL     420  52 EE      501  35 1/X
178  35 1/X    259  03  3      340  40  40    421  76 LBL     502  65  ×
179  95  =     260  01  1      341  65  ×      422  53  (      503  43 RCL
180  35 1/X    261  05  5      342  43 RCL     423  65  ×      504  47  47
181  42 STO    262  69 DP      343  41  41    424  43 RCL     505  95  =
182  36  36    263  04  04     344  65  ×      425  34  34    506  35 1/X
183  55  ÷     264  43 RCL     345  43 RCL     426  95  =      507  55  ÷
184  43 RCL    265  37  37     346  42  42    427  42 STO     508  02  2
185  30  30    266  65  ×      347  55  ÷      428  44  44    509  55  ÷
186  95  =     267  43 RCL     348  43 RCL     429  69 DP     510  89  π
187  71 SBR    268  38  38     349  37  37    430  06  06    511  95  =
188  24 CE     269  95  =      350  55  ÷      431  91 R/S     512  69 DP
189  32 X:T    270  69 DP      351  53  (      432  76 LBL     513  06  06
190  76 LBL    271  06  06     352  01  1      433  19 D'      514  91 R/S
191  35 1/X    272  91 R/S     353  75  -      434  01  1      515  76 LBL
192  73 RC*    273  04  4      354  43 RCL     435  05  5      516  45 YX
193  01  01    274  01  1      355  42  42    436  00  0      517  01  1
194  77  GE    275  01  1      356  55  ÷      437  02  2      518  00  0
195  42 STO    276  04  4      357  43 RCL     438  69 DP     519  42 STO
196  69 DP     277  00  0      358  31  31    439  04  04    520  34  34
197  21  21    278  01  1      359  54  )      440  02  2      521  01  1
198  61 GTO    279  69 DP      360  95  =      441  65  ×      522  03  3
199  35 1/X    280  04  04     361  71 SBR     442  89  π      523  42 STO
200  76 LBL    281  43 RCL     362  24 CE      443  65  ×      524  01  01
201  42 STO    282  37  37     363  32 X:T     444  43 RCL     525  92 RTN
202  65  ×     283  65  ×      364  76 LBL     445  45  45    526  76 LBL
203  43 RCL    284  43 RCL     365  43 RCL     446  95  =      527  24 CE
204  34  34    285  35  35     366  73 RC*     447  42 STO     528  42 STO
205  95  =     286  85  +      367  01  01    448  46  46    529  33  33
206  42 STO    287  43 RCL     368  22 INV     449  65  ×      530  01  1
207  35  35    288  25  25     369  77  GE     450  53  (      531  00  0
208  69 DP     289  95  =      370  44 SUM     451  43 RCL     532  32 X:T
209  06  06    290  42 STO     371  69 DP     452  43  43    533  76 LBL
210  91 R/S    291  39  39     372  31  31    453  35 1/X     534  34 FX
211  76 LBL    292  69 DP      373  61 GTO     454  85  +      535  43 RCL
212  16 A'     293  06  06     374  43 RCL     455  43 RCL     536  33  33
213  02  2     294  91 R/S     375  76 LBL     456  44  44    537  55  ÷
214  04  4     295  76 LBL     376  44 SUM     457  35 1/X     538  01  1
215  01  1     296  17 B'      377  65  ×      458  54  )      539  00  0
216  05  5     297  48 EXC     378  43 RCL     459  35 1/X     540  95  =
217  69 DP     298  45  45     379  34  34    460  95  =      541  77  GE
218  04  04    299  48 EXC     380  95  =      461  35 1/X     542  25 CLR
219  43 RCL    300  41  41     381  42 STO     462  69 DP     543  92 RTN
220  27  27    301  48 EXC     382  43  43    463  06  06    544  76 LBL
221  55  ÷     302  40  40     383  69 DP      464  91 R/S     545  25 CLR
222  43 RCL    303  91 R/S     384  06  06    465  01  1      546  42 STO
223  36  36    304  04  4      385  91 R/S     466  05  5      547  33  33
224  95  =     305  01  1      386  03  3      467  00  0      548  01  1
225  42 STO    306  01  1      387  05  5      468  03  3      549  00  0
226  37  37    307  04  4      388  00  0      469  69 DP     550  49 PRD
227  69 DP     308  00  0      389  03  3      470  04  04    551  34  34
228  06  06    309  01  1      390  69 DP     471  43 RCL     552  61 GTO
229  91 R/S    310  06  6      391  04  04    472  32  32    553  34 FX
230  04  4     311  05  5      392  71 SBR     473  85  +      554  76 LBL
231  01  1     312  69 DP      393  45 YX      474  43 RCL     555  54  )
232  01  1     313  04  04     394  43 RCL     475  28  28    556  00  0
233  05  5     314  01  1      395  43  43    476  95  =      557  23 LNX
234  01  1     315  85  +      396  65  ×      477  65  ×      558  91 R/S
235  07  7     316  43 RCL     397  53  (      478  43 RCL     559  00  0
236  69 DP     317  40  40     398  43 RCL     479  46  46
237  04  04    318  95  =      399  31  31    480  95  =
238  43 RCL    319  65  ×      400  55  ÷      481  35 1/X
239  32  32    320  43 RCL     401  43 RCL     482  69 DP
240  85  +     321  39  39     402  42  42    483  06  06
241  43 RCL    322  95  =      403  75  -      484  91 R/S
242  35  35    323  42 STO     404  01  1      485  76 LBL
243  95  =     324  42  42     405  54  )      486  10 E'
```

```
          0.75      UE
           7.5      UA
          2200.     RA
           0.5     USAT
           0.6      UBE
           100.      F



            24.     UB
          1800.     R3
           100.     R4
 .0075757576       IC
9.606060606        UCE
 .0727731864       PC
1.357575758        UBO
1.493333333        UBO'
          6800.    R1
        100000.    R2
 0.00000025        C1
 .0000003979       C2
8038128.439         F
```

Bild 3. Typischer Ausdruck beim Errechnen der Werte für einen Nf-Vorverstärker. Es wurde mit folgenden Angaben gerechnet: $\hat{U}_e = 0{,}75$ V, $\hat{U}_a = 7{,}5$ V, $R_a = 2200\ \Omega$, $f_u = 100$ Hz (untere zu übertragende Frequenz)

Hier wird eine Problemlösung gleich in drei unterschiedlichen Versionen angeboten: Für den programmierbaren Taschenrechner HP-41, für Pascal- und schließlich für Basic-Computer.

Hans-Georg Joepgen

„Notenassistent" sorgt für Zeugnistransparenz

Es gab einmal eine Zeit, da setzten Lehrer am Schuljahresende mit Hilfe der Noten aus einigen schriftlichen Arbeiten und aus ein paar Kritzelzahlen im Taschenkalender, ansonsten aber ziemlich über den Daumen und weitgehend nach „pädagogischem Gefühl", die Zeugnisnoten ihrer Schützlinge fest. So geht's heute nicht mehr; seit Zeugnisse im Konfliktfall einer Überprüfung durch Verwaltungsgerichte unterliegen, müssen Lehrer in der Lage sein, auch im Kreuzverhör durch ausgefuchste Rechtsanwälte jederzeit nachweisen zu können, welche Schülerleistungen mit welchem Gewicht beim Zustandekommen der Noten beteiligt waren. Dies bedeutet vor allem eins: Penibles Listenführen, tagelange Rechenarbeiten – und das jedesmal dann, wenn ein Zeugnistermin im Jahr naht: Viele tausend Einzelnoten hat ein Lehrer im Schnitt zu weit über 100 Zeugnisnoten zu verdichten.

Fehler beim Rechnen mit Taschenrechnern oder gar von Hand können böse Konsequenzen haben: Ungerechtigkeiten gegenüber Schülern, blamable Niederlagen in Prozessen, die der nächsten Beförderung gewiß nicht dienlich sind. Eine Situation also, die nach dem Einsatz von Mikrocomputern verlangt – und zwar von druckenden Mikrorechnern, die auch eine nachträgliche Kontrolle der eingegebenen Daten erlauben. So sind denn an gewissen Schulen mit fortschrittlichen Lehrern (und entsprechenden Etats) seit einiger Zeit die Computer aus dem Informatikunterricht der reformierten Oberstufe längst nicht nur im Unterricht eingesetzt, sondern tun auch Dienst beim Notenmachen.

Meist arbeiten diese Maschinen nach hausgeschneiderten Programmen, die auf die speziellen Bedürfnisse eines einzigen Faches zugeschnitten sind und überdies mit den Gebräuchen der Nachbarschule absolut nicht unter einen Hut zu bringen sind: Mit der „Portabilität", der Austauschbarkeit dieser Programme, sieht's also schlecht aus. Der Autor entwickelte aus diesem Grunde ein kleines Programmpaket, das sich – jeweils ohne Eingriffe ins betreffende Programm selbst und auch völlig ohne Programmierkenntnisse des Bedieners – flexibel unterschiedlichen Forderungen anpaßt: Rechnungen nach dem Punktesystem, Rechnungen mit Hilfe prozentualer Gewichtung der Posten aus der Kriterienliste, freie Gewichtung mit Hilfe von Proportionalfaktoren schließlich. Dabei werden gewisse Forderungen der Kultusbehörden, wie zum Beispiel das Abwerfen der zweiten Nachkommastelle in bestimmten Fällen, ebenso berücksichtigt wie die Tatsache, daß der Anteil der Halbjahresnote beim Zustandekommen der Schuljahresnote von Bundesland zu Bundesland und darüber hinaus von Schulart zu Schulart wechselt.

Der hierfür anfänglich verwendete Algorithmus, zuerst mit einem PET erprobt, wurde im Laufe zweier Schuljahre schrittweise verbessert und schließlich in drei Programmversionen übertragen, die auf PASCAL-Computern (Dialekte Standard-Pascal und UCSD-Pascal), auf Microsoft-Maschinen (CBM, ITT-2020, Apple-II, AIM-65, PC-100 und TRS-80) sowie auf dem Taschencomputer HP-41 C laufen.

Anforderungen an alle drei Versionen

Hauptforderung an die Programme war, neben dem Wunsch nach Flexibilität, saubere Dokumentation des Rechenganges – dazu kam noch die Notwendigkeit, den Benutzer präzis zu führen: Ihm wegen der großen Datenmenge jederzeit anzuzeigen, welche Teilnote zu welchem Schüler gerade einzugeben ist, so daß bei Arbeitsunterbrechungen Rechenfortsetzung ohne den Zwang zu Aufschrieben über den Stand der Rechnung möglich wurden. Weiterhin mußte der Tatsache Rechnung getragen werden, daß auch bei nicht vorliegenden Teilnoten, wenn der Schüler also gefehlt hatte, die Verteilung der Gewichte erhalten blieb. Die *Tabelle* zeigt eine schematische Formulierung des allen drei Programmen zugrundeliegenden Verfahrens zur Problemlösung.

Tabelle: Lösungsweg

(allen drei Fassungen von „Notenassistent" gemeinsam)

1. Einlesen: Schüleranzahl
   Anzahl der Teilnoten
   Jahresendrechnung: Ja/nein
2. Wenn Jahresendrechnung:
   1. Einlesen: Gewicht der Altnote,
      Gewicht der Neunote
3. Für jede Einzelnote:
   1. Einlesen: Namen der Note,
      Gewicht der Note
4. Für jeden Schüler
   „Summe" und „Teiler" auf Null setzen; sodann:
5. Für jede Einzelnote:
   1. Einlesen: Nummer und Namen der Note ausgeben,
      Wert der Note erfragen (0: Note liegt nicht vor)
      Produkt aus Note und Gewicht in „Summe" aufaddieren;
      Wenn Note ungleich Null:
      „Gewicht" in „Teiler" aufaddieren
      Sodann:
      „Summe" durch „Teiler" dividieren, hierbei nicht runden, alle Nachkommastellen außer der ersten abwerfen; Ergebnis (Halbjahresnote) ausgeben
6. Wenn Jahresendrechnung:
   Letzte Halbjahresnote erfragen, mit Hilfe von „Altgewicht" und „Neugewicht" wie oben Jahresendnote errechnen und ausgeben.

Wenn für letzten Schüler erledigt:
Job-Ende melden,
Programmschluß.

```
          PRP ""
       01♦LBL "N"
       02 FIX 1
       03 CLRG
       04 " ? "
       05 ASTO 28
 06 "EINZELNOTEN:"
       07 PROMPT
       08 12
       09 X<>Y
       10 X≠Y?
       11 GTO "N"
       12 STO 27

       13♦LBL 01
       14 1
       15 ST+ 00
       16 "NAME :"
       17 AON
       18 PROMPT
```

```
 19 ASTO IND 00
    20 AOFF
    21 XEQ 05
 22 "GEWICHT :"
    23 PROMPT
  24 STO IND 00
    25 XEQ 06
    26 RCL 27
    27 RCL 00
    28 X<>Y
    29 X>Y?
    30 GTO 01

    31♦LBL 02
    32 0
    33 STO 25
    34 STO 26
    35 STO 00

    36♦LBL 03
    37 1
```

```
    38 ST+ 00
    39 CLA
    40 "NOTE "
 41 ARCL IND 00
    42 ":"
    43 PROMPT
    44 STO 29
    45 XEQ 05
 46 RCL IND 00
    47 ST* 29
 48 RCL IND 00
    49 ST+ 25
    50 XEQ 06
    51 RCL 29
    52 ST+ 26
    53 RCL 00
    54 RCL 27
    55 X>Y?
    56 GTO 03
 57♦LBL 04
    58 RCL 26
```

```
 59 RCL 25
 60 /
 61 10
 62 *
 63 INT
 64 10
 65 /
 66 STOP
 67 GTO 02

 68♦LBL 05
 69 12
 70 ST+ 00
 71 RTN

 72♦LBL 06
 73 12
 74 ST- 00
 75 .END.
```

Bild 1. Programm-Auflistung zur Taschencomputer-Version

```
NOTE AUFS 1 :
                2,7     RUN
NOTE AUFS 2 :
                3,0     RUN
NOTE AUFS 3 :
                3,5     RUN
NOTE DIKT 1 :
                1,2     RUN
NOTE DIKT 2 :
                1,5     RUN
NOTE REFER :
                2,5     RUN
NOTE COLLOQ :
                4,5     RUN
NOTE MUENDL :
                2,7     RUN
NOTE HEFTF :
                1,0     RUN

                2,6     ***
```

Bild 2. So führt der HP-41 C Buch über alle Eingaben und fragt mit Klartext-Abkürzungen Schülerleistungen ab

```
PROGRAM NOTENASSISTENT;

CONST Z = 10.0;
VAR JAHRESENDE                  : BOOLEAN;
    PROMPT                      : CHAR;
    SNUMMER,SANZAHL,NNUMMER,
    NANZAHL,ALTGEWICHT,
    NEUGEWICHT                  : INTEGER;
    TEILNOTE, NOTE, ENDNOTE,
    SUMME, TEILER, ALTNOTE      : REAL;
    NAME : ARRAY [1..15] OF STRING [15];
    GEWICHT : ARRAY [1..15] OF INTEGER;

PROCEDURE LEERRAUM (N:INTEGER);
    VAR I:INTEGER;
     BEGIN
      FOR I:= 1 TO N DO
       WRITE (' ');
     END;

PROCEDURE STRICH (N:INTEGER);
    VAR I:INTEGER;
     BEGIN
      FOR I:=1 TO N DO WRITE('-')
     END;

PROCEDURE TABDRUCK;
    BEGIN WRITELN
     (' + SCHUELER # ',SNUMMER:2);
     WRITE ('   ');
     STRICH (13); WRITELN;
    END;

PROCEDURE TRENNUNG;
    BEGIN
     STRICH(40);WRITELN;
    END;

PROCEDURE HOLEN;
    BEGIN WRITELN;
     SUMME:= 0; TEILER:= 0;
     FOR NNUMMER:=1 TO NANZAHL DO
      BEGIN
       WRITE
       (NNUMMER:2,':',NAME[NNUMMER]:15);
       WRITE ('.-');
       WRITE (' TEILNOTE?   ':11);
       READ (TEILNOTE);
       IF TEILNOTE  <> 0 THEN
        BEGIN
         TEILER:=
         TEILER + GEWICHT [NNUMMER];
         SUMME:= SUMME +
         GEWICHT [NNUMMER] * TEILNOTE
        END;
      END; WRITELN
    END;

PROCEDURE QUOTENFRAGE;
    BEGIN
     WRITE
     (' +++  GEWICHT DER ALTNOTE?   ');
     READLN (ALTGEWICHT);
     WRITE
     (' +++  GEWICHT DER NEUNOTE?   ');
     READLN (NEUGEWICHT); WRITELN;
    END;

PROCEDURE RECHNEN;
    BEGIN
     WRITELN;
     IF TEILER = 0 THEN TEILER:= 1E-10;
     NOTE:=
     (TRUNC (Z * SUMME / TEILER)) / Z;
     WRITELN(' N O T E : ':16,NOTE:5:1);
     WRITELN
    END;

PROCEDURE SCHLUSSRECHNUNG;
    BEGIN
     WRITE (' +++ ALTNOTE ?   ');
     READLN  (ALTNOTE);
     TEILER:= ALTGEWICHT + NEUGEWICHT;
     SUMME :=
     ALTNOTE * ALTGEWICHT +
     NOTE * NEUGEWICHT; WRITELN;
     ENDNOTE:=
     TRUNC(Z * SUMME/TEILER)/Z;
     WRITE ('E N D N O T E  : ');
     WRITELN (ENDNOTE:2:1); WRITELN;
    END;

(* HAUPTPROGRAMM *)

BEGIN
  PAGE(OUTPUT);
  WRITELN ('         NOTENASSISTENT');
  WRITELN ('         ==============');
  WRITELN;
  WRITELN
  ('EIN PROGRAMM VON H.-G.  JOEPGEN');
  WRITELN
  ('+++++++ COPYRIGHT 1979 ++++++++');
  STRICH(31); WRITELN; WRITELN;
  WRITE
  ('  +++ JAHRESRECHNUNG (J/N) ? ');
  READLN (PROMPT);
  JAHRESENDE:= PROMPT= 'J';
  WRITE ('  +++ WIE VIELE SCHUELER?');
  LEERRAUM (4);
  READLN  (SANZAHL);
  WRITE ('  +++ ZAHL DER TEILNOTEN?');
  LEERRAUM (4);
  READLN  (NANZAHL); WRITELN;
  IF JAHRESENDE THEN QUOTENFRAGE;

(* NAMEN UND GEWICHTUNG EINLESEN *)

FOR NNUMMER:= 1 TO NANZAHL DO
 BEGIN
  WRITE ('NAME DER TEILNOTE ');
  WRITE ('NUMMER ',NNUMMER:2,': ');
  READLN (NAME[NNUMMER]);
  LEERRAUM (19);
  WRITE ('GEWICHT:  ');
  READLN (GEWICHT[NNUMMER]);
  WRITELN
 END;

(* NOTENRECHNUNG *)

FOR SNUMMER:= 1 TO SANZAHL DO
 BEGIN; TRENNUNG;
  WRITELN;
  TABDRUCK; HOLEN; RECHNEN;
  IF JAHRESENDE THEN SCHLUSSRECHNUNG;
 END;

(* SCHLUSS-SEGMENT *)

WRITELN
('  ===  J O B  B E E N D E T .-');
TRENNUNG
 END.
```

Bild 3. Die Pascal-Fassung von ‚Notenassistent': Ohne Tricks und Eleganz, aber dafür transparent formuliert

Die Taschenrechner-Variante

Am wenigsten Komfort bietet naturgemäß das Arbeiten mit dem Taschencomputer, wiewohl auch der HP-41 C die Teilnoten unter Nennung ihres Namens abfragt und an besonders kritischen Stellen im Programmablauf durch Plausibilitätskontrollen vor gewissen Bedienfehlern schützt.

Das Erprobungsgerät war mit einem Speicher-Erweiterungsmodul, einem Kartenadapter und einem Mini-Thermodrucker ausgerüstet. Der Name der Teilnote wurde, um abrollende Zeichendarstellung zu vermeiden, auf sechs Zeichen beschränkt. Der Benutzer hat, in Abhängigkeit vom Umfang seiner Teilnotenliste, bei Initiierung des Programms die Aufteilung zwischen Programm- und Datenspeicher von Hand seinen Bedürfnissen anzupassen, sofern mehr als 12 Einzelkriterien pro Endnote ins Spiel kommen sollen (*Bild 1* und *Bild 2*).

Die Pascal-Version

Als problemangepaßte Lösung hätte sich in Pascal, da die Zahl der Einzelnoten bei Programmstart nicht feststeht, die Einführung dynamischer Variabler und der Einsatz des Datentyps „Record" angeboten. Da sich derlei Kunstgriffe, wiewohl elegant, effizient und zur Demonstration der großen Leistungsfähigkeit von Pascal sicherlich sehr geeignet, dem Verständnis von Lesern ohne Pascal-Erfahrung zweifellos entzogen hätten, wird hier eine etwas archaische Version von „Notenassistent" vorgestellt, die 15 Teilnoten handhabt, deren Namen wiederum 15 Zeichen umfassen dürfen. Implementiert wurde das Programm auf einem Apple-II-Plus, der mit der Language-Karte, zwei Disk-Laufwerken, einem Drucker Centronics 779 sowie einer seriellen Schnittstellenkarte zum Betrieb einer (als Schrott erstandenen) Uralt-Console unbekannten Typs verwendet wurde. Von allen drei Versionen lief, natürlich, das Pascal-Programm am schnellsten (*Bild 3*).

Am komfortabelsten: Die Basic-Version

Eine (an praktisch vorkommenden Verhältnissen gemessen) unbeschränkte Menge von Teilnoten verkraftet die Basic-Version von „Notenassistent", geschrieben im Microsoft-Dialekt PALSOFT und erprobt auf einer mit 48 kByte RAM ausgerüsteten Maschine ITT 2020, die nachträglich durch neue ROMs mit residentem Fließkomma-Basic ausgerüstet wurde (*Bild 4* und *Bild 5*). Das Programm ist unverändert einsetzbar auf allen Apple-II- und ITT-

```
                NOTENASSISTENT
                ==============

EIN PROGRAMM VON H. - G. JOEPGEN,
+++++++ COPYRIGHT 1979 ++++++++

   +++   JAHRESRECHNUNG (J/N) ? J
   +++   WIE VIELE SCHUELER?    28
   +++   ZAHL DER TEILNOTEN?    12

   +++   GEWICHT DER ALTNOTE?   1
   +++   GEWICHT DER NEUNOTE?   2

NAME DER TEILNOTE #  1:  AUFSATZ NR 1
         GEWICHT      :  4

NAME DER TEILNOTE #  2:  AUFSATZ NR 2
         GEWICHT      :  4

NAME DER TEILNOTE #  3:  AUFSATZ NR 3
         GEWICHT      :  5

NAME DER TEILNOTE #  4:  DIKTAT NR 1
         GEWICHT      :  2

NAME DER TEILNOTE #  5:  DIKTAT NR 2
         GEWICHT      :  2

NAME DER TEILNOTE #  6:  LUECKENTEST
         GEWICHT      :  3

NAME DER TEILNOTE #  7:  WORTFELDARBEIT
         GEWICHT      :  2

NAME DER TEILNOTE #  8:  REFERAT 1
         GEWICHT      :  1

NAME DER TEILNOTE #  9:  REFERAT 2
         GEWICHT      :  2

NAME DER TEILNOTE # 10:  COLLOQUIUM
         GEWICHT      :  3

NAME DER TEILNOTE # 11:  MUENDLICH
         GEWICHT      :  6

NAME DER TEILNOTE # 12:  HEFTFUEHRUNG
         GEWICHT      :  1
```

Bild 5. In der Eröffnungsschleife holt sich die Basicfassung des Programms alle Angaben, die zur korrekten und selbsttätigen Anpassung an die verschiedenen Verfahren zur Notenberechnung erforderlich sind

```
10  REM  NOTENASSISTENT
20  REM  ==============
30  REM
40  HOME : REM FUER PET STATTDES-
    SEN: PRINTCHR$(147)
50  PRINT " ": PRINT " ": PRINT "
                NOTENASSISTENT"
60  PRINT "          ============
    =="
70  PRINT " ": PRINT "EIN PROGRAM
    M VON H. - G. JOEPGEN,"
80  PRINT "+++++++ COPYRIGHT 1979
     ++++++++": PRINT " ": PRINT
    " "
90  REM
100  REM STARTWERTE EINLESEN
110  REM --------------------
120  REM
130  INPUT "   +++   JAHRESRECHNU
    NG (J/N) ? ";PR$
140  IF PR$ <  > "J" AND PR$ <  >
    "N" GOTO 130
150  INPUT "   +++   WIE VIELE SC
    HUELER?    ";SA
160  INPUT "   +++   ZAHL DER TEI
    LNOTEN?    ";NA: PRINT " "
170  IF PR$ = "N" GOTO 200
180  INPUT "   +++   GEWICHT DER
    ALTNOTE?   ";AG
190  INPUT "   +++   GEWICHT DER
    NEUNOTE?   ";NG
200  DIM NA$(NA),GE(NA)
210  REM
220  REM KRITERIENLISTE EINLESEN
230  REM ------------------------
240  REM
250  PRINT " "
260  FOR NN = 1 TO NA:NN$ =  STR$
    (NN): IF  LEN (NN$) < 2 THEN
    NN$ = " " + NN$
270  PRINT "NAME DER TEILNOTE # "
    NN$";  ";
280  INPUT " ";NA$(NN):NA$(NN) =
    " " + NA$(NN)
290  IF  LEN (NA$(NN)) < 15 THEN
    NA$(NN) = " " + NA$(NN): GOTO
    290
300  PRINT  TAB( 9)" ";
310  INPUT "GEWICHT      :  ";GE(
    NN)
320  PRINT " ": NEXT
330  REM
340  REM HAUPTSCHLEIFE
350  REM -------------
360  REM
370  HOME : REM FUER PET STATT-
    DESSEN: PRINTCHR$(147)
380  PRINT " ": PRINT " ": FOR SN
     = 1 TO SA: PRINT " ": PRINT
    " SCHUELER "SN: PRINT " ----
    -------": PRINT " "
390  GOSUB 480: REM ARBEITSWERTE
    HOLEN UND AUFSUMMIEREN
400  GOSUB 580: REM ABRECHNUNG
    DURCHFUEHREN
410  IF PR$ = "N" GOTO 430
420  GOSUB 640: REM JAHRESRECH- N
    UNG DURCHFUEHREN
430  NEXT SN: PRINT " ": PRINT "
    + JOB BEENDET.  BYE. -": END
440  REM
450  REM ARBEITSWERTE
460  REM ------------
470  REM
480 S = 0:TR = 0: FOR NN = 1 TO N
    A:NN$ = " " +  STR$ (NN)
490  IF  LEN (NN$) < 6 THEN NN$ =
    " " + NN$: GOTO 490
500  PRINT NN$". "NA$(NN);
510  INPUT "; NOTE: ";TN: IF TN =
    0 GOTO 530
520 TR = TR + GE(NN):S = S + TN *
    GE(NN)
530  NEXT NN: RETURN
540  REM
550  REM ABRECHNUNG
560  REM ----------
570  REM
580 N =  INT (10 * S / TR) / 10
590  PRINT : PRINT "   +++   NOTE
    :    "N: RETURN
600  REM
610  REM SCHLUSSRECHNUNG
620  REM ---------------
630  REM
640  INPUT "   +++   ALTNOTE? ";T
    N
650  IF TN = 0 THEN EN = N: GOTO
    670
660 EN =  INT (10 * (TN * AG + N *
    NG) / (AG + NG)) / 10
670  PRINT : PRINT "   ===   ENDN
    OTE: "EN: RETURN
```

Bild 4. Das ist ‚Notenassistent' für Rechner wie ITT 2020 und Apple II, Hinweise zum Umbau in eine PET-Version sind im Listing selbst eingefügt

```
SCHUELER 8
----------

  1... AUFSATZ NR 1: NOTE: 3.2
  2... AUFSATZ NR 2: NOTE: 3
  3... AUFSATZ NR 3: NOTE: 2
  4.... DIKTAT NR 1: NOTE: 4.5
  5.... DIKTAT NR 2: NOTE: 4
  6.... LUECKENTEST: NOTE: 2
  7. WORTFELDARBEIT: NOTE: 1.5
  8..... REFERAT 1: NOTE: 2
  9..... REFERAT 2: NOTE: 0
 10..... COLLOQUIUM: NOTE: 0
 11..... MUENDLICH: NOTE: 3.5
 12... HEFTFUEHRUNG: NOTE: 1
+++   NOTE:    2.8
+++   ALTNOTE? 4
===   ENDNOTE: 3.2
```

```
SCHUELER 9
----------

  1... AUFSATZ NR 1: NOTE: 2
  2... AUFSATZ NR 2: NOTE: 2.5
  3... AUFSATZ NR 3: NOTE: 3
  4.... DIKTAT NR 1: NOTE: 2.2
  5.... DIKTAT NR 2: NOTE: 3
  6.... LUECKENTEST: NOTE: 4.5
  7. WORTFELDARBEIT: NOTE: 3
  8..... REFERAT 1: NOTE: 1.5
  9..... REFERAT 2: NOTE: 1.2
 10..... COLLOQUIUM: NOTE: 2
 11..... MUENDLICH: NOTE: 3
 12... HEFTFUEHRUNG: NOTE: 5
+++   NOTE:    2.7
+++   ALTNOTE? 3
===   ENDNOTE: 2.8
```

Bild 6. Ergebnisausdruck der Pascal- und der Palsoft-Basic-Fassung von ‚Noten-Assistent'. Eine Null in der Eingabekolonne bedeutet, daß der Schüler wegen Krankheit fehlte oder der Klassenarbeit aus anderen Gründen aus dem Weg ging: Alle drei Programmversionen passen in solchen Fällen die Gewichtung der Einzelnoten den neuen Verhältnissen gerade so an, wie das die Schulaufsichtsbehörden wünschen

2020-Maschinen, wobei jedoch gegebenfalls der entsprechende Applesoft- oder Palsoft-Interpreter noch von Band oder Scheibe zuvor zu laden ist. „Notenassistent" in dieser Version ist so geschrieben, daß es auch auf dem PET und auf CBM-Maschinen läuft, wenn die im Listing selbst angegebenen Änderungen vorgenommen und einige Zeilen gesplittert werden: Der PET mag ja in aller Regel keine Zeilen mit mehr als 80 Zeichen. *Bild 6* zeigt den Ausdruck bei Pascal und Basic.

Zur Erprobung auf der ebenfalls in immer mehr Schulen eingesetzten Maschine TRS-80 gab's leider keine Gelegenheit: Im „Zugriffsbereich" des Verfassers steht augenblicklich kein solcher Rechner zur Verfügung. Laut Handbuch-Angaben zu diesem Computer dürfte jedoch auch der TRS-80 durchaus in der Lage sein, mit Hilfe von „Notenassistent" die Tage (und Nächte!) des Zeugnis-Schreibens fortan auf einen halben Nachmittag zusammenschrumpfen zu lassen. Eine Bemerkung zum Schluß: Bisher war immer nur die Rede von Lehrern. Aber „Notenassistent" eignet sich natürlich in allen drei Versionen auch hervorragend für Schüler, die zwischendurch mal wissen wollen, wie denn die Lage so ist... ■

TI-59-Mehrfachplotter

Die Druckerfunktion OP 07 des TI 59/58 hat den Nachteil, daß jeweils nur ein Punkt pro Zeile geplottet werden kann. Mit dem folgenden Unterprogramm kann man vier Punkte pro Zeile plotten. Allerdings wird statt einem Sternchen eine Null gedruckt. Fallen nun n Funktionswerte auf die gleiche Druckerposition, so wird „(n-1)" an der betreffenden Stelle ausgedruckt.

Beispiel: Fallen drei Werte auf die gleiche Position, so erscheint statt einer „0" eine n-1 = 3-1 = „2". Dieser Vorteil ist ein zufälliges Nebenprodukt, welches auf der nicht vermeidbaren Addition der Codezahlen beruht. Die Funktionswerte müssen zwischen 0 und 19 liegen und können auch „krumme" Werte annehmen. Die Anzahl der Funktionen (1...4) muß im Datenregister 9 vor jedem Unterprogrammaufruf neu abgespeichert werden. Die Funktionswerte sind in den Registern 15...18 festzuhalten. Aufruf des Unterprogramms mit SBR 000.

Sollte die Funktionsanzahl größer als die Anzahl der Funktionswerte sein, so wird die überzählige Funktion als y = 0 (x-Achse) interpretiert. Sollten vier Funktionen und eine x-Achse zu zeichnen sein, so darf man ausnahmsweise „5" in Register 9 speichern. Ansonsten wurde, um wertvolle Programmzeilen zu sparen, auf Tests innerhalb des Unterprogramms verzichtet, ob etwaige Grenzen überschritten wurden. Die Speicher R9–R18 sind belegt.

Die Druckzeit pro Zeile beträgt etwa 4...11 Sekunden, je nach Funktionsanzahl. Das Beispiel zeigt y = sin x, y = cos x, y = 0,5 und y = −0,5. (Die Rechenzeit betrug für *Bild 2* fünf Minuten.) *Wolfgang Umlauf*

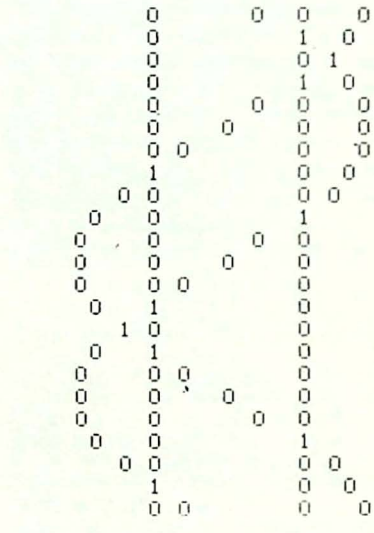


Bild 2. Beispielausdruck zweier Sinusfunktionen

```
000  25 CLR | 022  52 EE  | 044  97 DSZ | 066  08  8  | 088  13  13 | 110  69 OP
001  42 STO | 023  32 X:T | 045  09  09 | 067  71 SBR | 089  01  1  | 111  03  03
002  11  11 | 024  04  4  | 046  00  00 | 068  01  01 | 090  49 PRD | 112  43 RCL
003  42 STO | 025  77  GE | 047  13  13 | 069  10  10 | 091  11  11 | 113  14  14
004  12  12 | 026  00  00 | 048  61 GTO | 070  44 SUM | 092  49 PRD | 114  69 OP
005  42 STO | 027  79  79 | 049  00  00 | 071  12  12 | 093  12  12 | 115  04  04
006  13  13 | 028  09  9  | 050  89  89 | 072  97 DSZ | 094  49 PRD | 116  69 OP
007  42 STO | 029  77  GE | 051  02  2  | 073  09  09 | 095  13  13 | 117  05  05
008  14  14 | 030  00  00 | 052  08  8  | 074  00  00 | 096  49 PRD | 118  92 RTN
009  01  1  | 031  65  65 | 053  71 SBR | 075  13  13 | 097  14  14 |
010  04  4  | 032  01  1  | 054  01  01 | 076  61 GTO | 098  43 RCL | 119  75  -
011  42 STO | 033  04  4  | 055  10  10 | 077  00  00 | 099  11  11 | 120  02  2
012  10  10 | 034  77  GE | 056  44 SUM | 078  89  89 | 100  22 INV | 121  65  ×
013  01  1  | 035  00  00 | 057  13  13 | 079  08  8  | 101  58 FIX | 122  32 X:T
014  44 SUM | 036  51  51 | 058  97 DSZ | 080  71 SBR | 102  69 OP  | 123  95  =
015  10  10 | 037  03  3  | 059  09  09 | 081  01  01 | 103  01  01 | 124  22 INV
016  73 RC* | 038  08  8  | 060  00  00 | 082  10  10 | 104  43 RCL | 125  28 LOG
017  10  10 | 039  71 SBR | 061  13  13 | 083  44 SUM | 105  12  12 | 126  52 EE
018  58 FIX | 040  01  01 | 062  61 GTO | 084  11  11 | 106  69 OP  | 127  22 INV
019  00  00 | 041  10  10 | 063  00  00 | 085  97 DSZ | 107  02  02 | 128  52 EE
020  52 EE  | 042  44 SUM | 064  89  89 | 086  09  09 | 108  43 RCL | 129  92 RTN
021  22 INV | 043  14  14 | 065  01  1  | 087  00  00 | 109  13  13 | 130  00  0
```

Bild 1. Programm zum Plotten mehrerer Kurven mit dem Drucker PC-100

Zur Vorbereitung von Flügen nach Sichtflugregeln (VFR-Flüge) benötigt der Pilot zur Navigation den zu steuernden Kurs, um von einem Ausgangspunkt zum Zielpunkt zu gelangen, die Entfernung zwischen beiden Punkten, die Flugzeit, den Treibstoffverbrauch und diese Daten unter Berücksichtigung des Einflusses des Windes.

Dr. Hermann Schnell

Flugvorbereitung

Da die Flugvorbereitung vor Beginn des Fluges am Boden durchgeführt wird, kann hierfür auch ein Programm für Rechner wie der PET-2001, die im Flugzeug nicht mitgeführt werden können, gute Dienste leisten.

Das Programm „Flugvorbereitung" in PET-Basic (3655 Bytes) berechnet Kurse und Entfernungen für Flugwege auf den zur Vorbereitung von VFR-Flügen geeigneten Luftfahrtkarten. Die Algorithmen wurden so einfach wie möglich gewählt. Dabei wurde bewußt auf eine, ohnehin nicht erreichbare, absolute Genauigkeit verzichtet. Es ist sinnlos, Entfernungen genauer zu berechnen, als sie der Karte entnommen werden können, und Kurse genauer zu ermitteln, als sie in Sportflugzeugen mit Hilfe des Magnetkompasses zu erfliegen sind. Für über die Pole führende Kurse müssen die Pole als Etappenziele gewählt werden. Das Programm ist aber für Kursberechnungen in den Polargebieten ohnehin nicht geeignet, da dort die berechneten Kurse in den meisten Fällen nicht die kürzeste Verbindung zwischen zwei Punkten darstellen. Dies ist kein Nachteil, da die Polargebiete von Sportflugzeugen nicht beflogen werden.

Das Programm fordert als erstes die Daten des verwendeten Flugzeugtyps an: Eigengeschwindigkeit in Knoten (Seemeilen pro Stunde), ausfliegbarer Tankinhalt in Litern, Treibstoffverbrauch in Litern pro Stunde, sowie die Zahl der Etappen, die berechnet werden sollen (Zeilen 160–190).

Die Berechnung des Windeinflusses wird als Option angeboten, um Ø Eingaben zu vermeiden (Zeile 200). Nun werden für den Startflughafen und jedes Etappenziel die Koordinaten angefordert, die in Grad, Minuten und Sekunden eingegeben werden. Hierbei ist wichtig, daß westliche Längen und südliche Breiten mit negativem Vorzeichen eingegeben werden (Zeilen 230–360 -GGG, -MM, -SS).

Für jede Etappe wird weiterhin der Wind nach Richtung in Grad und Stärke in Knoten angefordert. Westliche Mißweisungen werden mit negativen Vorzeichen eingegeben. Die Berechnung der Entfernung zwischen Ausgangspunkt und Etappenziel sowie des rechtweisenden Kurses basiert auf der Voraussetzung, daß Längenkreise und Breitenparallele rechte Winkel bilden. Dies ist auch bei Luftfahrtkarten in Kegelprojektion, für Entfernungen, wie sie Etappen bei Flügen mit Sportflugzeugen entsprechen, mit ausreichender Genauigkeit der Fall.

In dem rechtwinkligen Dreieck aus Ausgangspunkt, Breitenparallel und Etappenziel entspricht die Strecke zwischen Ausgangspunkt und Schnittpunkt des Längenkreises mit dem Breitenparallel in Seemeilen der Breitendifferenz der beiden Punkte in Minuten.

Für die Strecke vom Schnittpunkt bis zum Etappenziel berechnen wir die Abweitung mit dem Produkt aus der Längendifferenz der beiden Punkte in Minuten mit dem Cosinus der Mittelbreite (arithmetisches Mittel der Breiten von Ausgangspunkt und Etappenziel).

Die Entfernung auf dem Flugweg in Seemeilen wird mit dem Pythagoräi-

```
100 CLR:PRINT"s          *****FLUGVORBEREITUNG*****"
110 REM COPYRIGHT(C)1979 DR.HERMANN                SCHNELL
120 PRINT"     FUER VFR FLUEGE MIT SPORTFLUGZEUGEN":PRINT
130 REM FUER FLUGKARTEN IN KEGELPRO-               JEKTION
140 REM*BEI KURSEN UEBER DIE POLE                  MUESSEN
150 REM DIE POLE ALS ETAPPENZIELE                  GEWAEHLT WERDEN*
160 INPUT"EIGENGESCHWINDIGKEIT(KN)";VE
170 INPUT"AUSFLIEGBARER TANKINHALT(L)",T
180 INPUT"TREIBSTOFFVERBRAUCH (L/H)";VT
190 INPUT"WIEVIELE ETAPPEN",E
200 INPUT"SOLL DER WINDEINFLUSS BERUECKSICHTIGT   WERDEN(J/N)";FT$
210 PRINT:PRINT
220 IFE>9THEN:GOSUB1280
230 INPUT"NAME DES STARTFLUGHAFENS",E$(0)
240 PRINT"KOORDINATEN DES STARTFLUGHAFENS"
250 PRINT"GGG,MM,SS LAENGE OST+  WEST-
260 INPUT"BREITE:";K(1),K(2),K(5)
270 INPUT"LAENGE:";K(3),K(4),K(6)
280 IFK(1)>90ORK(1)<-90THEN240
290 IFK(3)>180ORK(3)<-180THEN240
300 FORI=1TOE
310 IFE=1THEN920                              BREITE NORD+ SUED-"
320 PRINT:PRINT"NAME ETAPPENZIEL"I;
330 INPUTE$(I)
340 PRINT"KOORDINATEN:ETAPPENZIEL"I
350 INPUT"BREITE:";KB(I),KC(I),KS(I)
360 INPUT"LAENGE:";KD(I),KE(I),KT(I)
370 IFFT$="N"THEN:LV=0:WR(I)=0:WS(I)=0:GOTO410
380 PRINT"WIND AUF ETAPPE"I"
390 PRINT"-RICHTUNG(GRAD),STAERKE(KN)-";:INPUTWR(I),WS(I)
400 IFWR(I)>360ORWR(I)<0THEN1350
410 PRINT"MISSWEISUNG-WEST-                                        OST+";
420 INPUTM(I)
430 IFKB(I)>90ORKB(I)<-90ORKD(I)>180ORKD(I)<-180THEN:GOTO1320
440 PRINT"SIND ALLE EINGABEN KORREKT?(J/N)"
450 GETZ$:IFZ$=""THEN450
460 IFZ$="N"THEN:PRINT:GOSUB1330  GoTo 1330
470 NEXTI
480 KF(0)=K(1)*60+K(2)+(K(5)/60)
490 KG(0)=K(3)*60+K(4)+(K(6)/60)
500 FORI=1TOE
510 KF(I)=KB(I)*60+KC(I)+(KS(I)/60)
520 KG(I)=KD(I)*60+KE(I)+(KT(I)/60)
```

```
530 DB(I)=KF(I)-KF(I-1):X(I)=ABS(DB(I))
540 DL(I)=KG(I)-KG(I-1):Y(I)=ABS(DL(I))
550 IFABS(DL(I))>10800THEN:GOSUB1380
560 PRINT:PRINTSPC(10)"*****":PRINT
570 PRINT"ETAPPE"I;
580 PRINTE$(I-1)"-"E$(I)
590 PRINT
600 PRINT"FLUGSTRECKE (NM):";
610 Z(I)=ABS((KF(I)+KF(I-1)))/2
620 L(I)=SQR(X(I)↑2+(Y(I)*COS(Z(I)/60*π/180))↑2)
630 PRINTINT(L(I)*10+.5)/10
640 IFINT(L(I))<3THEN700
650 S=(KF(I)-KF(I-1))/L(I)
660 RK=(-ATN(S/SQR(-S*S+1))+1.5708)*180/π
670 IFDL(I)=>0THEN970
680 IFDL(I)=<0THEN1000
690 GOSUB1010
700 NEXTI
710 IFE>1THEN740
720 GOTO870
730 END
740 FORI=1TOE
750 LG=LG+L(I)
760 FG=FG+FT(I)
770 TG=TG+TV(I)
780 NEXTI
790 PRINT:PRINT
800 PRINT"GESAMTFLUGSTRECKE(NM):";INT(LG+.5)
810 FI=INT(FG):FK=(FG-FI)*60
820 PRINT"GESAMTFLUGZEIT:"FI"STUNDEN" INT(FK+.5)"MINUTEN"
830 PRINT"GESAMTTREIBSTOFFVERBRAUCH(L):";INT(TG+.5)
840 PRINTINT(TG/T*100)"% DES TANKINHALTS"
850 PRINT:PRINTSPC(10)"****"
860 IFE>9THEN:END
870 PRINT:PRINT"WOLLEN SIE WEITERE KURSE MIT DEN
880 PRINT"EINGEGEBENEN FLUGZEUGDATEN BERECHNEN     (J/N)?"
890 GETY$:IFY$=""THEN890
900 IFY$="J"THEN:LG=0.FG=0.TG=0:PRINT:GOTO190
910 END
920 PRINT"NAME ZIELFLUGHAFEN:";
930 INPUTE$(I)
940 PRINT"KOORDINATEN ZIELFLUGHAFEN:"
950 GOTO350
960 RETURN
970 IFS=1THEN:RK=0
980 IFS=-1THEN:RK=180
990 PRINT"RECHTWEISENDER KURS(GRAD):"INT(RK+.5):GOTO690
1000 PRINT"RECHTWEISENDER KURS(GRAD):"360-INT(RK+.5):RK=360-RK:GOTO690
1010 IFFT$="N"THEN:VG=VE:GOTO1100
1020 W=WR(I)-180-RK
1030 SS=SIN(W*π/180)*WS(I)
1040 YY=SS/VE
1050 LU=ATN(YY/SQR(-YY*YY+1))*180/π
1060 VG=COS(LU*π/180)*VE+COS(W*π/180)*WS(I)
1070 LV=INT(LU+.5)*-1
1080 PRINT"LUVWINKEL(GRAD):";LV
1090 PRINT
1100 PRINT"GRUNDGESCHWINDIGKEIT(KN):";INT(VG+.5)
1110 PRINT
1120 FT(I)=L(I)/VG:FH=INT(FT(I)):FM=(FT(I)-INT(FT(I)))*60
1130 PRINT"FLUGZEIT"FH"STUNDEN" INT(FM+.5)"MINUTEN"
1140 SK=INT(RK+LV-M(I)+.5)
1150 IFSK<0THEN:SK=SK+360
1160 IFSK>360THEN:SK=SK-360
1170 PRINT"STEUERKURS(GRAD):"SK
1180 TV(I)=VT*FT(I)
1190 PRINT"TREIBSTOFFVERBRAUCH(L):",INT(TV(I)+.5)
1200 PRINTINT(TV(I)*100/T)"% DES AUSFL.TREIBSTOFFS"
1210 IFE>1THEN:GOSUB1240
1220 RETURN
1230 END
1240 PRINT:PRINT
1250 PRINT"RUFEN SIE WEITERE ANGABEN DURCH DRUECKEN EINER TASTE AB"
1260 GETX$:IFX$=""THEN1260
1270 RETURN
1280 DIMKB(E),KC(E),KS(E),KD(E),KE(E),KT(E),WR(E)
1290 DIMWS(E),M(E),KF(E),KG(E),DB(E),DL(E),L(E),Z(E),FT(E),TV(E)
1300 DIME$(E),X(E),Y(E)
1310 RETURN
1320 PRINT"FALSCHE EINGABE DER KOORDINATEN BEI     ETAPPE"I
1330 I=I-1
1340 GOTO470
1350 PRINT"FALSCHE EINGABE DES WINDES                BEI ETAPPE"I
1360 I=I-1
1370 GOTO470
1380 A=SGN(DL(I))
1390 DL(I)=(21600-ABS(DL(I)))*-A
1400 Y(I)=ABS(DL(I))
1410 RETURN
```

Bild 1. Programm zur Flugvorbereitung, geschrieben im PET-Dialekt des Microsoft-Basic. In Zeile 460 muß es GOTO statt GOSUB heißen

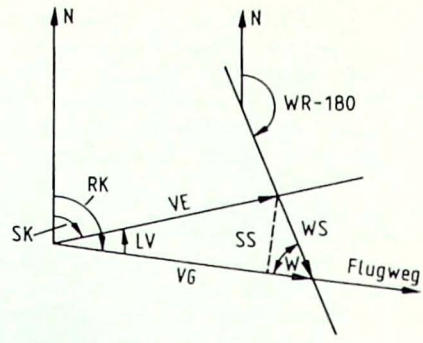


Bild 2. Winddreieck

schen Lehrsatz berechnet (Zeilen 480–630). Der Winkel zwischen Längenkreis und Flugweg in Grad wird mit der Cosinus-Funktion ermittelt. Da PET den Arcus-Cosinus nicht direkt berechnen kann, geschieht dies über den Arcus-Tangens. Der PET rechnet mit Winkeln im Bogenmaß (rad), deshalb müssen alle Eingabegrößen in Grad (Altgrad) mit $\pi/180$ und die in rad erhaltenen Ergebnisse mit $180/\pi$ multipliziert werden (Zeilen 650–660).

Leider berechnet der PET bei trigonometrischen Funktionen statt Null oft eine, wenn auch sehr kleine, Zahl und betrachtet umgekehrt sehr kleine Zahlen als Null. Um bei kleinen Entfernungen auf dem Flugweg einen „DIVISION BY ZERO ERROR" zu vermeiden, wird bei Entfernungen von weniger als 3 Seemeilen die Kursberechnung ausgelassen (Zeile 640). Über die Nord-Süd-Richtung (Kurse 0 Grad und 180 Grad) wird mit Hilfe des Quotienten aus Breitendifferenz und Entfernung zwischen östlichen und westlichen Kursen mit Hilfe der Längendifferenz entschieden. Längendifferenz positiv bedeutet östliche Kurse, Längendifferenz negativ westliche Kurse (Zeilen 670–680, 970–1000). Das letztere führt dazu, daß bei Flugwegen, die über den 180. Längengrad führen, Kurse berechnet werden, die auf einem Großkreis um die Erde herum führen. Durch ein Unterprogramm wird sichergestellt, daß auch hier der kürzeste Flugweg errechnet wird (Zeilen 550, 1380–1400).

Der Einfluß des Windes wird mit Hilfe des Winddreiecks berechnet (*Bild 2*). Da dieses nur in Ausnahmefällen rechtwinklig ist, teilen wir es mit der Höhe SS in zwei rechtwinklige Dreiecke. Durch Windrichtung und rechtweisenden Kurs ist der Winkel W bekannt (Zeile 1020). Mit dessen Hilfe wird die Strecke SS (Zeile 1030), der Luvwinkel (Zeilen 1050, 1070) und die Grundgeschwindigkeit (Zeile 1060) berechnet. Alle anderen Rechnungen sind trivial.

Durch Dialogverkehr mit dem Rechner soll das Programm die Benutzung nach Möglichkeit erleichtern. Um bei

der Berechnung mehrerer Etappen bei falschen Eingaben eine erneute Eingabe aller Daten zu vermeiden, wird nach jeder Etappe zur Kontrolle der letzten Eingabe aufgefordert und Gelegenheit zur Korrektur gegeben (Zeilen 440–460, 1330–1340). Auch bei der Flugvorbereitung ist Vertrauen gut, aber Kontrolle besser! Die errechneten Kurse und Entfernungen sollten deshalb grundsätzlich mit Hilfe der Karte kontrolliert werden.

Anm. d. Red.: Dieser Artikel soll kein Einführungskurs für Flugnavigation sein. Dem interessierten Leser wird empfohlen, sich Spezialliteratur darüber zu besorgen, z. B. das „Taschenbuch für das fliegende Personal". ■

TI-59 plottet Funktionen

Das nachfolgend beschriebene TI-59-Programm gibt Datenpunkte auf dem Drucker PC-100 als Graph aus. Dabei wird automatisch der Maßstab so gewählt, daß der vorhandene Platz von 20 Zeichen pro Zeile optimal ausgenutzt wird.

In der in *Bild 1* aufgelisteten Programmversion sind maximal 50 Dateneingaben zulässig. Werden mehr gebraucht, so läßt sich dies leicht verwirklichen, indem man die Speicherbereichsverteilung z. B. auf 239.89 ändert, die Adressen der Hinweisregister (50...54) z. B. auf 85...89 verschiebt und die Schritte 20 und 21 ändert.

Der Gebrauch ist recht einfach: Nach dem Eingeben des Programms drückt man E; damit ist der Rechner vorbereitet und die Dateneingabe kann beginnen. Hierzu wird der Wert eingegeben und die Taste A gedrückt; danach wird die Anzahl der bisher eingegebenen Datenpunkte angezeigt. Sind alle Werte gespeichert, so wird nach Betätigen der Taste B der Graph ausgedruckt. Ferner werden der kleinste und größte Wert sowie der Mittelwert angegeben (*Bild 2*).

Rainer Maaß

000	76	LBL	050	55	÷	100	54	)	150	24	CE
001	15	E	051	01	1	101	00	0	151	43	RCL
002	47	CMS	052	22	INV	102	42	STO	152	55	÷
003	22	INV	053	44	SUM	103	50	IND	153	22	INV
004	86	STF	054	50	IND	104	04	4	154	49	PRD
005	01	01	055	43	RCL	105	07	7	155	56	T
006	91	R/S	056	52	EE	106	04	4	156	69	OP
007	76	LBL	057	32	X↔T	107	07	7	157	05	05
008	11	A	058	73	RC*	108	04	4	158	69	OP
009	87	IFF	059	50	IND	109	07	7	159	00	00
010	01	01	060	22	INV	110	04	4	160	04	4
011	23	LNX	061	77	GE	111	07	7	161	05	5
012	72	ST*	062	00	00	112	04	4	162	03	3
013	50	IND	063	66	PAU	113	07	7	163	00	0
014	01	1	064	42	STO	114	69	OP	164	02	2
015	44	SUM	065	52	EE	115	01	01	165	04	4
016	50	IND	066	43	RCL	116	69	OP	166	03	3
017	43	RCL	067	53	(	117	02	02	167	01	1
018	50	IND	068	32	X↔T	118	69	OP	168	69	OP
019	32	X↔T	069	73	RC*	119	03	03	169	04	04
020	04	4	070	50	IND	120	69	OP	170	43	RCL
021	09	9	071	77	GE	121	04	04	171	53	(
022	22	INV	072	00	00	122	69	OP	172	69	OP
023	77	GE	073	76	LBL	123	05	05	173	06	06
024	23	LNX	074	42	STO	124	73	RC*	174	04	4
025	43	RCL	075	53	(	125	50	IND	175	05	5
026	50	IND	076	01	1	126	44	SUM	176	03	3
027	91	R/S	077	22	INV	127	56	)	177	00	0
028	76	LBL	078	44	SUM	128	75	-	178	01	1
029	23	LNX	079	50	IND	129	43	RCL	179	03	3
030	86	STF	080	43	RCL	130	53	(	180	04	4
031	01	01	081	50	IND	131	95	=	181	04	4
032	43	RCL	082	32	X↔T	132	55	÷	182	69	OP
033	50	IND	083	01	1	133	43	RCL	183	04	04
034	94	+/-	084	94	+/-	134	54	)	184	43	RCL
035	34	√x	085	22	INV	135	95	=	185	52	EE
036	33	x²	086	67	EQ	136	59	INT	186	69	OP
037	91	R/S	087	00	00	137	69	OP	187	06	06
038	76	LBL	088	55	÷	138	07	07	188	06	6
039	12	B	089	43	RCL	139	01	1	189	07	7
040	25	CLR	090	52	EE	140	44	SUM	190	69	OP
041	43	RCL	091	75	-	141	50	IND	191	04	04
042	00	00	092	43	RCL	142	43	RCL	192	43	RCL
043	42	STO	093	53	(	143	50	IND	193	56	T
044	52	EE	094	95	=	144	32	X↔T	194	69	OP
045	42	STO	095	55	÷	145	43	RCL	195	06	06
046	53	(	096	01	1	146	55	÷	196	69	OP
047	43	RCL	097	09	9	147	22	INV	197	00	00
048	50	IND	098	95	=	148	67	EQ	198	98	ADV
049	42	STO	099	42	STO	149	01	01	199	98	ADV
									200	98	ADV
									201	91	R/S

Bild 1. Programmauflistung zum Darstellen von Datenpunkten auf dem Drucker des TI-59

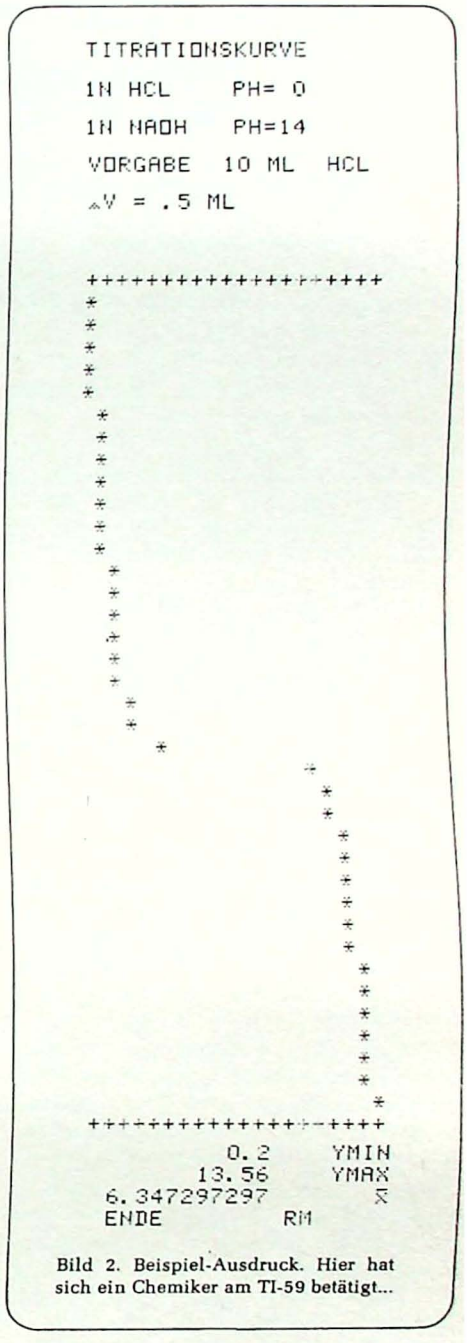


Bild 2. Beispiel-Ausdruck. Hier hat sich ein Chemiker am TI-59 betätigt...

Bevor der Mathematiker eine Funktion zeichnet, diskutiert er sie. Diese Arbeit wird um so zeitaufwendiger, je höher der Grad der Funktion ist. Das folgende Programm benötigt nur 3 KByte und liefert eine Kurvendiskussion für jede rationale Funktion.

Michael Konz

Kurvendiskussion in Basic

Das Programm (*Bild 1*) verarbeitet sowohl ganze als auch gebrochene rationale Funktionen beliebigen Grades. Die Koeffizienten werden beginnend mit dem Glied der höchsten Potenz nacheinander eingegeben.

Zuerst wird der Schnittpunkt mit der y-Achse berechnet, anschließend wird die Funktion auf Nullstellen untersucht. Ist sie quadratisch oder linear, so werden die Lösungen unmittelbar bestimmt.

In allen anderen Fällen wird das Newtonsche Näherungsverfahren benutzt. Die erste Näherung für x ist 0. Wenn eine Wurzel gefunden wurde, wird eine Polynomdivision durchgeführt und der Grad der Gleichung um eins erniedrigt. Diese Prozedur wird solange fortgesetzt, bis eine quadratische Gleichung vorliegt oder keine Nullstelle mehr gefunden werden kann.

Für Funktionen, deren höchstwertiges x einen ungeradzahligen Exponenten hat, wird die Rechnung erst abgebrochen, wenn eine Wurzel gefunden wurde, da eine solche Funktion immer eine Nullstelle hat.

Ist der höchste Exponent geradzahlig und wurde nach fünf Ansätzen mit fünf verschiedenen ersten Näherungswerten für x keine Nullstelle gefunden, wird ebenfalls abgebrochen.

Wem diese fünf Versuche nicht genügen oder zuviel sind (im ersten Fall steigt die Rechenzeit, im zweiten kann einem eine Wurzel entgehen), der mag die Zeile 845 entsprechend ändern. Mehrere verschiedene erste Näherungswerte für x sind aus folgendem Grund notwendig: Liegt zwischen dem ersten Näherungswert und der Nullstelle ein Extremum, dann versagt das Newtonsche Näherungsverfahren.

Im Anschluß an die Berechnung der Nullstellen erfolgt die Bestimmung der Pole beziehungsweise Unendlichkeitsstellen. Bei ganzen Funktionen wird dieser Schritt übersprungen.

Jetzt werden Extremstellen und Wendepunkte berechnet. Dazu werden die ersten beiden Ableitungen der Funktion gebildet. Da zur sicheren Bestimmung dieser Kurvenpunkte auch die höheren Ableitungen erforderlich wären, treten unter bestimmten Umständen Fehlinterpretationen auf, die aber ohne weiteres als solche zu erkennen sind. Es wird auch niemand auf die Idee kommen, daß die Funktion $y=x^6$ bei p(0;0) einen Sattelpunkt haben könnte.

Dieser kleine Fehler wurde in Kauf genommen, um das Programm durch

```
1 REM ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
2 REM ◆◆KURVENDISKUSSION◆◆
3 REM ◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆◆
4 REM MICHAEL KONZ,26.12.1979
9 DEFFNR(X)=INT(100◆X+.5)/100
10 INPUT"GEBROCHENE FUNKTION";A$:IFLEFT$(A$,1)<>"N"THENF=1
20 PRINT"GRAD ";:IFF=1THENPRINT"DES ZAEHLERS";:GOTO30
25 PRINT"DER FUNKTION";
30 INPUTNZ:IFF=1THENINPUT"GRAD DES NENNERS ";NN
40 PRINT:N=NZ:IFNN>NZTHENN=NN
45 NS=N:NH=NN+NZ-1:NK=2◆NN:NQ=NH+NK-1
50 DIMZ(NZ),N(NN),L(2◆N),F(3◆N),G(3◆N),H(NH),K(NK),Q(NQ)
60 IFF=1THENPRINT"ZAEHLER:"
80 FORI=NZTO0STEP-1:PRINT"A";I;"=";:INPUTZ(I):NEXT:PRINT
90 IFF=0THENN(0)=1:GOTO110
100 PRINT"NENNER:":FORI=NNTO0STEP-1:PRINT"A";I;"=";:INPUTN(I):NEXT:PRINT
105 REM --SCHNITTPUNKT MIT Y-ACHSE--
110 X=0:GOSUB970
115 IFYN=0THENPRINT"Y-ACHSE WIRD NICHT GESCHNITTEN":GOTO130
120 PRINT"SCHNITTPUNKT MIT Y-ACHSE BEI P( 0;";FNR(YZ/YN);")"
125 REM --NULLSTELLEN VON Y=F(X)--
130 PRINT:GOSUB995:GOSUB800
140 IFL=0THENPRINT"NULLSTELLEN EXISTIEREN NICHT":GOTO160
150 FORI=1TOL:PRINT"NULLSTELLE BEI X=";FNR(L(I)):NEXT
160 IFF=0THEN200
165 REM --POLE VON Y=F(X)--
170 PRINT:GOSUB996:GOSUB800
180 IFL=0THENPRINT"POLE EXISTIEREN NICHT":GOTO200
190 FORI=1TOL:PRINT"POL BEI X=";FNR(L(I)):NEXT
195 REM --1.ABLEITUNG Y' : H(X)/K(X)--
200 GOSUB995:GOSUB900:IFF=0THENFORI=0TONH:H(I)=G(I):NEXT:GOTO280
205 IFNZ=0THEN230
210 FORI=0TONZ-1:FORJ=0TONN:H(I+J)=H(I+J)+G(I)◆N(J):NEXTJ,I
230 IFNN=0THEN260
235 GOSUB996:GOSUB900
240 FORI=0TONN-1:FORJ=0TONZ:H(I+J)=H(I+J)-G(I)◆Z(J):NEXTJ,I
260 FORI=0TONN:FORJ=0TONN:K(I+J)=K(I+J)+N(I)◆N(J):NEXTJ,I
280 GOSUB997:GOSUB800
285 REM --2.ABLEITUNG Y'' : Q(X)--
290 GOSUB997:GOSUB900:IFF=0THENFORI=0TONQ:Q(I)=G(I):NEXT:GOTO350
295 IFNH=0THEN305
300 FORI=0TONH-1:FORJ=0TONK:Q(I+J)=Q(I+J)+G(I)◆K(J):NEXTJ,I
305 IFNK=0THEN350
310 N=NK:FORI=0TON:F(I)=K(I):NEXT:GOSUB900
320 FORI=0TONK-1:FORJ=0TONH:Q(I+J)=Q(I+J)-G(I)◆H(J):NEXTJ,I
345 REM --EXTREMSTELLEN--
350 PRINT:FORI=0TONQ:F(I)=Q(I):NEXT
360 N=NQ:H=0:IFL=0THEN428
370 FORJ=1TOL:X=L(J):GOSUB950
375 GOSUB970:IFABS(YN)<5E-4THEN427
380 IFYF=0THEN427
390 IFYF<0THENPRINT"MAXIMUM";
400 IFYF>0THENPRINT"MINIMUM";
425 PRINT" BEI P(";FNR(X);";";FNR(YZ/YN);")":H=1
427 NEXT
428 IFH=0THENPRINT"EXTREMSTELLEN EXISTIEREN NICHT"
429 REM --Y'' NULL SETZEN--
430 GOSUB800:N=NH:FORI=0TON:F(I)=H(I):NEXT:PRINT ▶
```

```
RUN
GEBROCHENE FUNKTION? Y
GRAD DES ZAEHLERS? 2
GRAD DES NENNERS ? 1

ZAEHLER:
A 2 =? 1
A 1 =? 3
A 0 =? 2

NENNER:
A 1 =? .5
A 0 =? 1.2

SCHNITTPUNKT MIT Y-ACHSE BEI
P( 0; 1.67 )


NULLSTELLE BEI X=-2
NULLSTELLE BEI X=-1

POL BEI X=-2.4

MAXIMUM BEI P(-3.15 ;-6.59 )
MINIMUM BEI P(-1.65 ;-.61 )

WENDEPUNKTE EXISTIEREN NICHT
```

Bild 2. Rechenbeispiel mit einer gebrochenen Funktion

die Berechnung der höheren Ableitungen nicht unnötig zu komplizieren.

Vorsicht ist auch geboten, wenn eine Funktion Lücken aufweist, wenn also Zähler und Nenner gleichzeitig Null werden. Das Programm erkennt derartige Lücken nicht, und bei der Berechnung der Extrema und Wendepunkte können wegen der unvermeidlichen Rechenungenauigkeiten verblüffende Ergebnisse auftauchen.

Desgleichen darf man sich nicht wundern, wenn dieselbe Nullstelle mehrmals angezeigt wird. Es handelt sich dann einfach um eine mehrfache Nullstelle, wie zum Beispiel bei $y = (x-3)^3$.

Bild 2 zeigt die Diskussion der Funktion $y = \frac{x^2+3x+2}{0{,}5x+1{,}2}$. Das Programmlisting entstand mit einem AIM-65 und einem Silent-700-Terminal. ■

Primfaktor-Zerlegung mit dem TI-59

Dieses Programm ist sehr nützlich, wenn man in der Digitaltechnik zum Beispiel verschiedene Taktzeiten aus einem Mutteroszillator erzeugen soll, oder einfach aus einer Kiste Quarze den passenden heraussuchen will, aus dem man möglichst einfach herunterteilen kann.

Das Programm ist universell mit und ohne Drucker zu verwenden. Mit Drukker ergibt sich nebenstehender Ausdruck, ohne Drucker werden die Primzahlen nacheinander im Pause-Mode angezeigt. Die letzte Zahl bleibt in der Anzeige stehen.

Das Programm ist sehr benutzerfreundlich aufgebaut, d.h. keine Vorbereitungstasten sind zu drücken. Die zu zerlegende Frequenz wird einfach nach Label A eingegeben, schon läuft alles ab.

Zur Auswertung: Paarweises Vorhandensein von 2en und 5en teilt man mit 10er-Zählern, weitere 2er mit 16er-Zählern, für die übrigen Primzahlen muß man die Zähler entsprechend setzen.

Dipl.-Ing. Fred-H. Heller

```
435 REM --WENDEPUNKTE--
440 H=0:IFL=0THEN480
450 FORJ=1TOL:X=L(J):GOSUB950
455 GOSUB970:IFABS(YN)<1E-3THEN475
460 IFYF=0THENPRINT"SATTELPUNKT";
465 IFYF<>0THENPRINT"WENDEPUNKT";
470 PRINT" BEI P(";FNR(X);";";FNR(YZ/YN);")":H=1
475 NEXT
480 IFH=0THENPRINT"WENDEPUNKTE EXISTIEREN NICHT"
490 END
790 REM --NULLSTELLEN SUBROUTINE--
800 L=0:IFN=0THENRETURN
805 FORI=NTO0STEP-1:IFF(I)<>0THENN=I:GOTO810
806 NEXT
810 H=0:A=0:IFN<3THEN920
815 GOSUB900
817 X=A:J=0:GOSUB950
820 GOSUB960:IFYG=0THENYG=1
830 X=X-YF/YG:GOSUB950:J=J+1
840 IFABS(YF)<1E-6THEN870
845 IFH<5THEN855
850 IFJ>30ANDINT(N/2)=N/2THENRETURN
855 IFJ<31THEN820
856 IFH=0THENA=1
860 A=A+-2:H=H+1:GOTO817
865 REM --NULLSTELLE GEFUNDEN--
870 L=L+1:L(L)=X
884 REM --POLYNOMDIVISION--
885 FORI=NTO1STEP-1
890 G(I-1)=F(I):F(I-1)=F(I-1)+X*G(I-1):NEXT:N=N-1
895 FORI=0TON:F(I)=G(I):NEXT:GOTO810
899 REM --1.ABLEITUNG VON F(X)--
900 IFN>0THENFORI=1TON:G(I-1)=F(I)*I:NEXT:RETURN
910 FORI=0TO3*N:G(I)=0:NEXT:RETURN
915 REM --GLEICHUNG QUADRATISCH ODER LINEAR--
920 IFF(2)<>0ANDN=2THEN927
921 REM --LINEAR--
922 IFF(1)<>0THENL=L+1:L(L)=-F(0)/F(1)
924 RETURN
926 REM --QUADRATISCH--
927 D=F(1)^2-4*F(2)*F(0):IFD<0THENRETURN
930 L=L+1:L(L)=(-F(1)-SQR(D))/2/F(2):IFD=0THENRETURN
940 L=L+1:L(L)=(-F(1)+SQR(D))/2/F(2):RETURN
945 REM --SUBROUTINES--
950 YF=0:FORI=0TON:YF=YF+F(I)*X^I:NEXT:RETURN
960 YG=0:FORI=0TON-1:YG=YG+G(I)*X^I:NEXT:RETURN
970 YN=0:FORI=0TONN:YN=YN+N(I)*X^I:NEXT
980 YZ=0:FORI=0TONZ:YZ=YZ+Z(I)*X^I:NEXT:RETURN
995 N=NZ:FORI=0TON:F(I)=Z(I):NEXT:RETURN
996 N=NN:FORI=0TON:F(I)=N(I):NEXT:RETURN
997 N=NH:FORI=0TON:F(I)=H(I):NEXT:RETURN
```

Bild 1. Basic-Programm in der AIM-65-Syntax (Microsoft-Basic) für die Kurvendiskussion – eine Wohltat für Mathematik-Studenten

▶ TI-58/59-Programm zur Zerlegung einer Zahl in ihre Primfaktoren. Links ist ein Rechenbeispiel zu sehen

```
000  76 LBL      042  01  1
001  11  A       043  07  7
002  98 ADV      044  03  3
003  99 PRT      045  01  1
004  42 STO      046  01  1
005  01  01      047  06  6
006  02  2       048  01  1
007  42 STO      049  07  7
008  00  00      050  69 DP
009  43 RCL      051  04  04
010  01  01      052  43 RCL
011  76 LBL      053  00  00
012  33 X²       054  69 DP
013  55  ÷       055  06  06
014  43 RCL      056  91 R/S
015  00  00      057  76 LBL
016  95  =       058  53  (
017  42 STO      059  43 RCL
018  03  03      060  00  00
019  75  -       061  66 PAU
020  01  1       062  99 PRT
021  95  =       063  43 RCL
022  67  EQ      064  03  03
023  34 fX       065  42 STO
024  22 INV      066  01  01
025  59 INT      067  69 DP
026  67  EQ      068  30  30
027  53  (       069  61 GTO
028  43 RCL      070  54  )
029  01  01  600600.
030  42 STO        2.
031  03  03       2.
032  76 LBL       2.
033  54  )        3.
034  69 DP        5.
035  20  20       5.
036  43 RCL       7.
037  03  03      11.
038  61 GTO      13.    ENDE
039  33 X²
040  76 LBL
041  34 fX
```

Linker Rand bei „LLIST“ und „LPRINT“

Bestimmt hat sich jeder TRS-80-Besitzer schon darüber geärgert, daß nahezu kein erhältlicher Drucker die Möglichkeit besitzt, einen linken Rand zu programmieren, der immer eingehalten wird. Ein solcher Rand ist z.B. dann notwendig, wenn man Listings u.ä. abheften will und wichtige Informationen nicht durch Löcher verstümmelt werden sollen.

In diesem Fall hilft nur ein Maschinenprogramm, da die Funktion „LLIST“ fest im Computer programmiert ist und man sich nicht (wie bei Basic-Befehlen „LPRINT“) durch Einfügen von Leerräumen in jedem Befehl helfen kann.

Bild 1 zeigt ein solches einfaches Maschinenprogramm. Es lädt in den Speicherbereich F000 bis F019 und ist daher in dieser Form direkt in 48-K-Geräten zu benutzen. Da keine absolute Adressierung vorkommt, kann es jedoch mit minimalen Änderungen in andere Speicherbereiche verschoben werden. Dazu muß lediglich Zeile 120 statt mit F000 mit der neuen Startadresse des Programms geladen und das Programm selbst in den neuen Bereich verschoben werden (Zeile 130 ändern). Für Anwendung in Basic-Programmen einfacher ist die in *Bild 2* gezeigte Version. Vor dem ersten Druckvorgang nach dem Einschalten des Computers sollte man dieses Programm einmal ablaufen lassen – danach ist das Maschinenprogramm im Speicher vorhanden.

Wie bei allen Maschinenprogrammen muß die „Memory Size“-Frage korrekt beantwortet werden, wenn der Computer eingeschaltet wird – welche Zahl Sie hier eingeben, hängt davon ab, in welchen Bereich das Maschinenprogramm verschoben wurde (was bei dem in Bild 2 gezeigten Programm durch Ändern der Zeile 10 möglich ist: erste Zahl ist das LSB der Startadresse, die zweite das MSB. Außerdem muß Zeile 60 mit der neuen Anfangs- und Endadresse geladen werden).

Beim Laden des Programms (ab Zeile 120) wird der Computer zunächst so programmiert, daß jeder „LLIST“- oder „LPRINT“-Befehl an das Maschinenprogramm „umgeleitet“ wird. Bei jedem Zeichen, das gedruckt werden soll, gelangt der Computer damit zur Stelle F000 im Speicher. Das zu druckende Zeichen befindet sich im C-Register und wird zunächst in den Akkumulator geladen (Zeile 137). Anschließend wird das Zeichen gedruckt, indem die normale Druckroutine des TRS-80 aufgerufen wird.

In Zeile 170 prüft der Computer, ob er gerade ein „Carriage Return“-Zeichen ausgegeben hat. Wenn ja, dann wurde die Druckzeile gerade beendet und als Vorbereitung für die nächste Zeile werden Leerräume ausgegeben (Zeilen 190 bis 240). Die Zahl der Leerräume (hier 5) wird in Zeile 210 festgelegt und kann beliebig geändert werden. Nachdem die Leerräume ausgegeben oder wenn kein „Carriage Return“ gegeben wurde, gelangt der Computer über Zeile 250 und 260 zurück ins Betriebssystem.

```
10 DATA 0,240
20 DATA 245,121,205,141,5,254,13,40,4,254,10,32,11,197,6,5,14,32
30 DATA 205,141,5,16,249,193,241,201
40 READ A,B
50 C=16422 : POKE C,A : POKE C+1,B
60 FOR C=-4096 TO -4071
70 READ A : POKE C,A
80 NEXT C
```

Bild 2. Dieses kurze Basic-Programm lädt das Maschinenprogramm von Bild 1 per POKE-Befehl in den TRS-80

```
                00100 ;£££ DRUCKERANSTEUERUNG MIT LINKEM RAND £££
4026            00110          ORG     4026H
4026 00F0       00120          DEFW    0F000H
F000            00130          ORG     0F000H
F000 F5         00135          PUSH    AF
F001 79         00137          LD      A,C
F002 CD8D05     00140          CALL    58DH
F005 FE0D       00170          CP      0DH
F007 2804       00180          JR      Z,SPACE
F009 FE0A       00182          CP      0AH
F00B 200B       00184          JR      NZ,ENDE
F00D C5         00190 SPACE    PUSH    BC
F00E 0605       00210          LD      B,5
F010 0E20       00215 S1       LD      C,32
F012 CD8D05     00220          CALL    58DH
F015 10F9       00230          DJNZ    S1
F017 C1         00240          POP     BC
F018 F1         00250 ENDE     POP     AF
F019 C9         00260          RET
0000            00270          END
00000 TOTAL ERRORS

ENDE    F018 00250    00184
S1      F010 00215    00230
SPACE   F00D 00190    00180
```

Bild 1. Maschinenprogramm in der Z-80-Syntax des TRS-80 zum Einhalten eines linken Randes beim Ausdruck

Lediglich zwei Probleme treten bei der Anwendung des Programms auf: Der linke Rand wird nur eingehalten, wenn keine Druckzeile so lang ist, daß sie in die nächste Zeile reicht. In der Praxis stellt dies kein Problem dar, da man aus Gründen der Übersichtlichkeit ohnehin derart lange Zeilen vermeidet. Das zweite Problem: Beim ersten Druckbefehl, nachdem das Programm geladen wurde, wird der Rand nicht eingehalten. Dieses Problem kann leicht beseitigt werden, indem als erster Befehl Leerräume gedruckt werden: LPRINT “ “.

Eine andere Möglichkeit ergibt sich, wenn ein Flag im Programm eingeführt wird, so daß beim ersten Durchlauf automatisch 5 Leerräume gegeben werden. Da ein solches Flag jedoch an einer festen Stelle definiert werden müßte, wurde dies hier nicht vorgesehen, um das Programm leicht verschiebbar zu machen.

Reinhard Wiesemann

Minotaurus per Computer

Das Programm stellt eine kleine Attraktion für jeden TRS-80-Besitzer dar. Es wurde auf einem TRS-80 (4 K RAM, Level-II-Basic) entwickelt und ist ein ideales Vorführprogramm für die ganze Familie. Es ermöglicht dem Benutzer, auf dem Schirm des Monitors ein Labyrinth bedienerfreundlich zu zeichnen. Das Labyrinth kann später mittels Programm abgeändert bzw. verbessert werden. Ist dieses fertiggestellt, so hat es der Benutzer (mit einem Start und Endpunkt versehen) an den Computer zu übergeben. Dieser steuert dann eine durch den Buchstaben „M" symbolisierte Maus völlig selbständig zum Endpunkt. Dies ist für den Zuschauer sehr lustig, überhaupt wenn sich die Maus des öfteren verirrt – doch immer findet sie ihr Ziel.

Das Programm gliedert sich in zwei Teile. Der erste ermöglicht es, bedienerfreundlich den Irrgarten zu erstellen, zu ändern und ihn dem Computer zu übergeben. Der zweite Teil verleiht der Maus ihre „Intelligenz" – er übernimmt die Steuerung der Maus! Dieser Teil – er umfaßt die Zeilen 2000 bis 3010 – ist jedoch für den Benutzer relativ uninteressant. Zeile 3010 ist eine Warteschleife, in welcher der Computer Instruktionen vom Benutzer erwartet.

Doch nun zum ersten Programmteil. Er wird vom Rechner direkt nach dem Einschalten und Starten durch RUN angelaufen. Nach Löschen des Schirmes sieht man den per Software blinkenden Cursor. Er befindet sich auf Position 0. Man kann nun mit dem Eingeben des Labyrinths beginnen. Sämtliche Tasten werden durch die INKEY-Funktion (PET: GET) ständig abgefragt, aber nur einige haben eine per Programm definierte Funktion – alle anderen Eingaben werden ignoriert. Ausgenommen ist lediglich die Break-Taste; sie ermöglicht zu jedem Zeitpunkt eine Unterbrechung des Programms. Im Labyrinth werden die Mauern durch einen „*" dargestellt, die Wege durch Leerzeichen. Der Startpunkt wird als „S" geschrieben, der Endpunkt als „END". Da der Abstand zwischen zwei benachbarten Zeichen in einer Zeile gleich dem zweier untereinanderliegender Zeichen sein soll, werden automatisch zwischen zwei Zeichen in einer Zeile zwei Leerzeichen geschoben. Daraus resultiert ein sauberes Bild. Folgenden Tasten ist eine Bedeutung zugeordnet:

Mit den durch Pfeile bezeichneten Tasten des TRS-80 kann man den Cursor nach links, rechts, oben und unten bewegen. Die Space-Taste gestattet das Schreiben eines Leerraums, und mit der Stern-Taste kann man „*" schreiben. Taste S schreibt das Symbol für die Startposition, Taste E das für die Endposition, und „Enter" (beim PET die Return-Taste) übergibt das auf dem Bildschirm stehende Labyrinth dem Computer. Bemerkenswert ist, daß hier die Schreibbefehle die Cursorposition nicht verändern; dies geschieht ausschließlich mit den Pfeil-Tasten. Wichtig ist, daß der Startpunkt auf der linken Seite des Labyrinths liegt, da die Maus sonst infolge der Suchtaktik das Labyrinth verläßt. Ist das Labyrinth gezeichnet, braucht man nur die „ENTER"-Taste zu drücken; die Maus beginnt nach kurzer Zeit ihren Lauf.

Hat die Maus ihr Ziel gefunden, signalisiert sie dies durch das im Ziel blinkende „M". Der Computer befindet sich nun in einer Warteschleife und erwartet neue Befehle. Die Tasten werden wiederum durch die INKEY-Funktion abgefragt. Folgende drei Tasten sind definiert:
C: Änderung des Labyrinths
E: Ende des Programms
R: Wiederholung der Suche der Maus im selben Labyrinth

Bei Eingabe von einem „C" erscheint der Cursor auf seiner alten Position wie vor Beginn der Suche. Das Programm befindet sich dann wieder in dem bereits am Anfang beschriebenen Teil und akzeptiert die Befehle 1 bis 9 wie bei einer Neueingabe. Ein Betätigen der „R"-Taste läßt die Maus von neuem auf Suche gehen. Ein „E" beendet das Programm und bewirkt den Rücksprung auf den Basic-Monitor.

Sinnvollerweise fügt man in Zeile 2010 noch GOTO 110 an, um einen Rücksprung in den Eingabemodus zu erreichen.

Und nun viel Spaß beim Eintippen des Labyrinths. Es lassen sich immer wieder mit Phantasie neue komplizierte Irrgärten schaffen, doch unsere Maus wird nie resignieren! *Jan Lichtermann*

```
10 DEFINTA-Z
100 CLS
110 PRINTCHR$(14);:FOR N=1TO50:A$=INKEY$:IFA$=""THENNEXTELSE120
111 PRINTCHR$(15);:FOR N=1TO50:A$=INKEY$:IFA$=""THENNEXTELSE120
115 GOTO110
120 A=ASC(A$)
140 IFA=91THENPRINTCHR$(27);
150 IFA=10THENPRINTCHR$(26);
160 IFA=9THENPRINTSTRING$(3,CHR$(25));
170 IFA=8THENPRINTSTRING$(3,CHR$(24));
180 IFA=58THENPRINT"x";CHR$(24);
190 IFA=32THENPRINT"   ";STRING$(3,CHR$(24));
200 IFA=69THENPRINT"END";STRING$(3,CHR$(24));
210 IFA=83THENPRINT"S";CHR$(24);
220 IFA=13THENPRINTCHR$(15);:GOTO2000
250 GOTO110
2000 REM SUCHEN
2010 FORN=15360TO16383:IFPEEK(N)=83THEN2050ELSENEXT
2050 H=0:CU=N
2060 ON H+1 GOSUB2200,2300,2400,2500
2070 X=PEEK(NN):IFX=69THEN3000
2080 IFX=42THEN H=H+1 ELSE H=H-1:POKENN,77:POKEN,32:N=NN
2090 H=H AND 3:GOTO2060
2200 NN=N+64:RETURN
2300 NN=N+3:RETURN
2400 NN=N-64:RETURN
2500 NN=N-3:RETURN
3000 POKEN,32:POKENN,77:POKENN+1,32:POKENN+2,32
3010 FORA=1TO100:A$=INKEY$:IFA$kg""THEN3020ELSENEXT:POKENN,32
3015 FORA=1TO100:A$=INKEY$:IFA$kg""THEN3020ELSENEXT:POKENN,77
3017 GOTO3010
3020 IFA$="C"ORA$="R"THENPOKENN,69:POKENN+1,78:POKENN+2,68
3025 POKECU,83
3030 IFA$="E"THENEND
3040 IFA$="R"THEN2000
3050 IFA$="C"THEN110ELSE3010
```

Dieses Programm erlaubt nicht nur das Schreiben komplizierter Irrgärten, sondern befähigt auch noch eine „Maus", sich darin hindurchzufinden. Zum Ausdruck diente eine Schreibmaschine mit dem in Heft 4/1980 der FUNKSCHAU beschriebenen Interface, weswegen „§" durch „S" zu ersetzen ist. Statt des Zeichens „*" wurde „x" ausgedruckt

Um kontinuierliche Funktionskurven darzustellen, ist der Mikrocomputer PET nur bedingt geeignet, da er einzelne Bildpunkte des Schirms nicht separat ansprechen kann, sondern immer nur Felder mit 8 x 8 Punkten. Mit einem kleinen Trick läßt sich trotzdem eine hinreichend gute Auflösung erreichen.

Peter-Mattias Oden

PET plottet Funktionen

Durch die zahlreichen Grafiksymbole, die der PET erzeugen kann, hat er auch die Möglichkeit, die 8 waagerechten Linien dieser Felder einzeln anzusprechen. Dadurch kann man eine maximale Auflösung von 200 x 40 „Strichen" erreichen, was für einen Überblick über Eigenschaft und Form der dargestellten Funktion völlig ausreicht.

Das hier beschriebene Programm hat eine etwas geringere Auflösung. Der besseren Übersichtlichkeit wegen werden noch ein Koordinatensystem, die Nullinie der Funktion (falls sie innerhalb des Wertebereiches liegt), sowie der anzugebende Definitionsbereich und Wertebereich angezeigt.

Die darzustellende Funktion steht in Zeile 230 des Programms (*Bild*) und kann jederzeit neu definiert werden.

BASIC-Programm zum Plotten mathematischer Funktionen auf dem Bildschirm des PET 2001. Der Ausdruck entstand mit einem Centronics-779-R-Drucker mit einem speziellen PET-Zeichengenerator

```
1 I=PEEK(630):IFI/2>0THEN PRINT"ACHTUNG----"I"LESEFEHLER! BITTE NEU EINLESEN":NEW
2 PRINT "s":PRINT"GRAPHISCHE DARSTELLUNG VON FUNKTIONEN
3 PRINT:PRINT:PRINT:PRINT"COPYRIGHT    :  PETER-M. ODEN
4 PRINT:PRINT"      DROSSELWEG 26"
5 PRINT:PRINT"    5042 ERFTSTADT-12"
6 FORN=1TO2000:NEXTN
10 DIMY(52):Y(0)=0
20 DIMZ(8):FORN=1TO8:READZ(N):NEXTN
25 DATA100,82,70,64,67,68,69,99
100 PRINT"s"
110 PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT:PRINT
120 PRINT"SIND SIE MIT DER BEDIENUNG VERTRAUT?":PRINT:PRINT:PRINT"                    (J/N)"
121 GETA$:IFA$=""GOTO121
122 IFA$="J"GOTO200
123 IFA$<>"N"GOTO10
124 PRINT"s":PRINT:PRINT:PRINT:PRINT:PRINT"DIESES PROGRAMM ERMOEGLICHT EINE GRAPHI-"
125 PRINT"SCHE DARSTELLUNG VON FUNKTIONEN AUF DEM ":PRINT"BILDSCHIRM IHRES"
126 PRINT:PRINT:PRINT:PRINT"uccci uccci uccccci"
127 :PRINT"bucih bu—k j-iu-k"
128 :PRINT"bj-kh bj—i   ||  "
129 :PRINT"bu—k bu—k   ||  "
130 :PRINT"b|    bj—i   ||  "
131 :PRINT"jk    j—k   jk  "
132 PRINT    :PRINT:PRINT:PRINT:PRINT:FORN=1TO3000:NEXTN
133 PRINT"♥":PRINT:PRINT:PRINT:PRINT"GEBEN SIE DIE GEWUENSCHTE FUNKTION  IN":PRINT:PRINT:PRINT
134 PRINT:PRINT:PRINT"  230  Y=FUNKTION (X)"
135 PRINT:PRINT:PRINT"BEISPIELSWEISE":PRINT:PRINT:PRINT"      230  Y=SIN(X)"
136 PRINT:PRINT:PRINT"BEENDEN SIE IHRE EINGABE MIT "
137 PRINT:PRINT"                RRETURN_"
138 PRINT:PRINT"UND STARTEN SIE DANN DAS PROGRAMM MIT"
139 PRINT:PRINT"       RUN       RRETURN_"
140 FORN=1TO9000:NEXTN
141 PRINT"♥":PRINT:PRINT"ZUSAETZLICH KOENNEN SIE DANN - UND AUCH"
142 PRINT:PRINT"NACH DEM PLOTTEN DER KURVEN DURCH DRUE-"
143 PRINT:PRINT"CKEN DER JEWEILIGEN TASTE VERSCHIEDENE":PRINT:PRINT"MOEGLICHKEITEN WAEHLEN.
144 FORN=1TO5000:NEXTN
150 PRINT"♥"
155 PRINT"  ◠"
160 PRINT"  |F|  :PLOTTEN DER GESPEICHERTEN"
165 PRINT"  ◡          FUNKTION"
170 PRINT"  ◠"
175 PRINT"  |D|  :PLOTTEN DER DIFFERENZIERTEN
180 PRINT"  ◡          FUNKTION"
181 PRINT"  ◠"
182 PRINT"  |I|  :PLOTTEN DER INTEGRIERTEN"
183 PRINT"  ◡          FUNKTION"
184 PRINT"  ◠"
185 PRINT"  |N|  :NEUSTART MIT VERAENDERTEM"
186 PRINT"  ◡      DEFINITIONSBEREICH"
187 PRINT"  ◠"
188 PRINT"  |E|  :ENDE DES PROGRAMMS BZW. EINGABE"
```

Nach Angabe des Bereiches, in dem die Funktion dargestellt werden soll, berechnet der PET die einzelnen Funktionswerte y(x) und sucht sich daraus den größten und den kleinsten heraus. Anschließend werden die Werte so umgerechnet, daß der PET weiß, welche Linie der 8x8-Matrix im Schirmbild er jeweils ansprechen muß.

Zusätzlich bietet das Programm die Möglichkeit, die einzelnen Kurven zu differenzieren, zu integrieren oder den darzustellenden Bereich zu verändern.

Zur Differentiation ist zu bemerken, daß, genau wie nach einer Integration, die ausgedruckten Wertebereiche mit Vorsicht zu genießen sind.
Bei jeder Integration entsteht ein konstanter Faktor, der sich bei weiteren Integrationen durch additive lineare oder quadratische Terme usw. bemerkbar macht.

Dem Fachmann mögen manche Teile dieses Programms als wenig elegant oder gar umständlich erscheinen. Das liegt daran, daß der Verfasser erst wenige Wochen mit dem PET gearbeitet hat und auch vorher keinerlei Erfahrungen mit BASIC hatte. Es wurde trotzdem konsequent versucht, mit den zur Verfügung stehenden Möglichkeiten des PET eine Anwendungsmöglichkeit aufzuzeigen, die außerhalb der üblichen Spiel-, Rechen- oder Sortierprogramme liegt.

Als Anregung noch einige interessante Funktionen:

Funktion	*von*	*bis*
y = SIN(x)	0	7
y = EXP(–x · x)	–2	2
y = x ↑ 3	–5	5
y = x · (x–2) · (x+1)	–1,5	2,5

Einige wichtige Variablen:
A = X-MIN
B = X-MAX
G = Y-MIN
F = Y-MAX
S = Zum Funktionswert gehörende Zeile
T = Zum Funktionswert gehörende Linie innerhalb der 8x8-Matrix

```
189 PRINT"  ◡          EINER NEUEN FUNKTION"
190 FORN=1TO9000:NEXTN
191 PRINT"♥":END
200 PRINT"♥":PRINT:PRINT"BITTE NENNEN SIE MIR
NUN- DURCH KOMMA   GETRENNT -
DIE GRENZEN DES"
202 PRINT"DEFINITIONSBEREICHES ; ":PRINT:PRINT
"  ALSO X-MIN UND   X-MAX ."
205 INPUTA,B
210 C=B-A:IFC<=0GOTO200
215 D=C/37
216 F=-(1E38)
217 G=1E38
220 FORN=0TO51
225 X=A+N*D
230 Y=(COS(X))↑2
235 Y(N+1)=Y
236 IFN>37GOTO260
240 IFY(N+1)>FTHENF=Y(N+1)
250 IFY(N+1)<GTHENG=Y(N+1)
260 NEXTN
270 PRINT"♥":PRINT"BITTE GEBEN SIE NUN
DIE GEWUENSCHTE DAR-STELLUNGSWEISE EIN"
280 PRINT"ALSO  F , D , I  ."
300 GETA$:IFA$=""THEN300
310 IFA$="E"GOTO1000
320 IFA$="D"GOTO500
330 IFA$="I"GOTO600
340 IFA$="F"GOTO700
350 IFA$="N"GOTO200
360 GOTO270
500 F=-(1E38)
501 G=1E38
505 FORN=1TO51
510 Y(N)=(Y(N+1)-Y(N))/D
515 IFN>38GOTO530
520 IFY(N)>FTHENF=Y(N)
525 IFY(N)<GTHENG=Y(N)
530 NEXTN
540 Y(52)=Y(51)
550 GOTO700
600 F=-(1E38):G=1E38
610 FORN=1TO51
620 Y(N+1)=Y(N)+Y(N+1)
625 IFN>38GOTO650
630 IFY(N)>FTHENF=Y(N)
640 IFY(N)<GTHENG=Y(N)
650 NEXTN
700 PRINT"♥"
710 FORN=0TO39
711 POKE(32808+N),99
712 NEXTN
713 FORN=0TO39
714 POKE(33728+N),100
715 NEXTN
716 FORN=0TO23
746 POKE(32808+N*40),101
747 NEXTN
750 FORN=0TO23
751 POKE(32847+N*40),103
752 NEXTN
760 POKE32808,79
765 POKE32847,80
770 POKE33728,76
775 POKE33767,122
776 POKE33738,76
777 POKE33748,76
778 POKE33758,76
779 POKE32818,79
780 POKE32828,79
781 POKE32838,79
782 E=F-G:IFE<.01THENG=G-
    (.01-E):F=F+(.01-E):E=F-G
783 H=SGN(G):I=SGN(F)
784 J=H+I:IFJ<>0GOTO792
785 K=(-G/E)*21:L=INT(K+.01):
    M=INT((K-L)*8+.02)+1
786 O=(L+1)*40:IFM<1THENM=1:
    IFM>8THENM=8
787 FORN=0TO39
788 P=N-O+33728
789 POKEP,Z(M)
790 NEXTN
791 POKE(33728-O),48
792 Y=F*1000:Y=INT(Y):Y=Y/1000
793 W=G*1000:W=INT(W):W=W/1000
794 PRINT"S";:PRINTA"( X (";B;
    :PRINT"     "W"( Y (";Y
800 Q=(E/170)*8
810 FORN=1TO38
820 R=(Y(N)-G)/Q
830 S=INT(R+.01)
840 T=INT((R-S)*8+.02) +1
845 IFT<1THENT=1
846 IFT>8THENT=8
850 U=33688+N-S*40
860 POKEU,Z(T)
870 NEXTN
880 GOTO 300
1000 PRINT"♥":END

READY.
```

Die einzelnen Unterprogramme sind:

1-191	Einführung
200-260	Berechnen der Funktionswerte im gegebenen Bereich
270-360	Abfrage der gewünschten Darstellungsart
500-550	Berechnen der Funktionswerte der differenzierten Kurve
600-650	Berechnen der Funktionswerte der integrierten Kurve
700-794	Schreiben des Koordinatensystems, der Nullinie und des Definitions- und Wertebereiches
800-880	Zeichnen der Funktion ■

AIM-65-Tips

Die praktisch identischen Mikrocomputer AIM-65 und PC-100 fanden inzwischen große Verbreitung. Hier deshalb einige Tips für den Umgang mit ihren Basic-Interpretern. Beim Abbruch eines Programmes nach STOP oder dem Drücken der Break-Taste (F1) ist es möglich, es später mit CONT fortzusetzen, ohne – im Gegensatz zu RUN – die Variablenwerte dabei zu löschen.

Vertippt man sich dummerweise vor dem CONT-Befehl, so handelt man sich zunächst einen SN ERROR (Syntaxfehler) ein, nach dem sich der CONT-Befehl nicht mehr anwenden läßt, weil nach CONT sofort CN ERROR ausgegeben wird. Doch läßt sich der Basic-Interpreter überlisten: Nach POKE 134,2 funktioniert CONT wieder, und das Programm kann auch nach vorhergehenden Fehlermeldungen oder nach Änderungen bestimmter Programmzeilen fortgesetzt werden.

Der Basic-Befehl NULL dient üblicherweise zum Einfügen von Fill-Zeichen, um nach einem CR-Zeichen einem Drucker genug Zeit für den Wagenrücklauf zu geben. AIM-65 und PC-100 decodieren den Befehl, können ihn aber nicht wirklich ausführen, sondern brechen das laufende Basic-Programm ab und geben FN ERROR aus.

Wie man in einem Basic-Programm Variablen, d.h. Texte und Zahlenwerte auf Kassette abspeichern und wieder laden kann, war bereits in Heft 1/1980 unter dem Titel „Basic-Texteditor" beschrieben. Es ist sogar möglich, mit dem AIM-Texteditor (im Monitor-ROM) Texte zu editieren, auf Kassette zu bringen und von Basic her zu laden. Aber: Die Texte dürfen keine Kommas und Doppelpunkte enthalten, da diese beiden Zeichen zur Trennung von Variablen dienen. Aus diesem Grund ist es leider (!) nicht möglich, ein komplettes Basic-Programm in String-Variablen einzulesen und per Basic-Editor zu editieren.

Fe.

Will man mit Hilfe eines Heimcomputers Dateien aufbauen, die laufend ergänzt und aktualisiert werden sollen und auf die man schnell zugreifen will, dann ist ein Floppy-Disk-Laufwerk ein unerläßliches Hilfsmittel. Der Fortschritt gegenüber dem Kassettenrecorder liegt nicht nur in der Datenübertragungsgeschwindigkeit, sondern vielmehr in der Möglichkeit, auf bestimmte Datenabschnitte (Records) gezielt zuzugreifen. Die folgenden Angaben gelten speziell für den TRS-80, sind aber grundsätzlich auch auf andere Heimcomputer übertragbar.

Rudolf Hofer

Dateiverarbeitung mit dem TRS-80

Sequentielle Dateien

Der TRS-80 bietet zwei grundsätzliche Möglichkeiten, Daten auf eine Diskette zu schreiben: als sequentielle Datei (*sequential file*) und als wahlfreie Datei (*random file*).

Die sequentielle Dateiverarbeitung unterscheidet sich kaum von der Datenspeicherung auf Kassette. Um Daten auf Diskette zu schreiben, benutzt man den Befehl PRINT# gefolgt von einer Datei-Nummer (File-Nr.) und den abzuspeichernden Variablen oder Ausdrükken, die jeweils durch Komma bzw. Strichpunkt getrennt sind. Ein Beispiel:

PRINT# 1, A; B; C$

speichert die numerischen Variablen A und B sowie die String-Variable C$ in der Datei Nummer 1 ab.

Werden zwei String-Variable hintereinander abgespeichert, dann sind sie durch ein zusätzliches Komma (in Anführungszeichen) zu trennen:

PRINT #1, A$; ", "; B$

Fehlt das Komma, und werden die beiden Variablen nur durch Strichpunkt getrennt, dann werden sie beim Einlesen zusammengezogen und als eine einzige interpretiert.

Eingelesen wird eine sequentielle Datei mit dem INPUT-Befehl. Um die Werte des ersten Beispiels den Variablen X, Y und Z$ zuzuweisen, wäre folgender Befehl nötig:

INPUT# 1, X, Y, Z$

Die Datei-Nummer, in diesem Fall 1, muß natürlich der im PRINT-Befehl entsprechen; sie darf eine Ganzzahl von 1...16 sein.

Vor dem Einschreiben und Auslesen muß die entsprechende Datei geöffnet und geschlossen werden. Das geschieht mit den Befehlen OPEN und CLOSE. Sie haben folgendes Format:

OPEN "Art", #File-Nr., "File-Name"

und

CLOSE File-Nr., File-Nr., ...

Entfällt bei CLOSE jeglicher Zusatz, werden alle Dateien geschlossen (auch bei den nachfolgend besprochenen wahlfreien Dateien). Beim OPEN-Befehl sind drei verschiedene *Arten* möglich. Sie werden durch die Buchstaben O, I und R ausgedrückt. Mit "O" wird eine sequentielle Datei eröffnet, in die nachfolgend geschrieben werden soll (*Output*); "I" wird verwendet, wenn man lesen will (*Input*). Wahlfreie Dateien werden in jedem Fall mit OPEN "R", # File-Nr., "File-Name" eröffnet – ganz gleich, ob man schreiben oder lesen will.

Während im Level-II-BASIC innerhalb von String-Variablen bestimmte Zeichen (z. B. Anführungszeichen und Kommas) nicht vorkommen dürfen, gilt diese Beschränkung im erweiterten Disk-BASIC nicht mehr: Mit dem Befehl LINE INPUT, der wie der normale INPUT-Befehl benutzt wird, weist man einer Variablen einen beliebigen Text zu. Er kann auch im Zusammenhang mit sequentiellen Dateien verwendet werden.

Um festzustellen, ob alle Daten gelesen wurden, ist die Funktion EOF vorhanden. Sie hat folgendes Format:
EOF (File-Nr.)
Sind alle Daten gelesen, liefert sie den Wert –1, andernfalls 0. Jede sequentielle Datei benötigt auf der Diskette mindestens 1280 Byte. Bei Bedarf werden jeweils weitere 1280 Byte belegt.

Wahlfreie Dateien

Wahlfreie Dateien bringen die eingangs erwähnte Möglichkeit, auf bestimmte Dateien gezielt zuzugreifen. Im Gegensatz zu sequentiellen Dateien speichern sie die Daten nicht im ASCII-Format, sondern in komprimierter Form. Außerdem kann nach dem OPEN-Befehl beliebig geschrieben oder gelesen werden.

Grundelement für jede Schreib/Lese-Operation ist ein "Record" (256 Byte). Jede Datei kann davon bis zu 329 enthalten. Dementsprechend wird jedem Record eine Nummer von 1...329 zugeordnet. Bevor ein Record auf die Diskette geschrieben wird, werden die Daten in einem Puffer abgelegt (obwohl der Puffer 256 Byte lang ist, dürfen nur 255 Byte benutzt werden). Mit dem FIELD-Befehl weist man verschiedenen Variablen ihren Platz im Puffer zu. Analog dazu liest man einen Record von der Diskette zuerst in den Puffer und entnimmt dann die Variablen-Werte aus den vorher vereinbarten Plätzen. Dazu ist der gleiche FIELD-Befehl nötig. Ein Beispiel soll das Verfahren verdeutlichen: In die Datei Nummer 2 mit dem Namen ADRESS sollen zunächst die String-Variablen VN$ (= Vorname, max. 10 Zeichen), NN$ (= Nachnahme, max. 15 Zeichen) und OR$ (= Ort, max. 15 Zeichen) eingeschrieben werden. Die Record-Nummer ist 65. Folgendes Programm erfüllt die Aufgabe:

```
100 OPEN "R", 2, "ADRESS"
110 FIELD 2, 10 AS VN$, 15 AS NN$, 15 AS OR$
120 LSET VN$ = "VORNAME" : LSET NN$ = "NACHNAME" : LSET OR$ = "ORT"
130 PUT 2,65
```

Zeile 110 bewirkt, daß den Variablen der Reihe nach 10, 15 und noch einmal 15 Byte zur Verfügung gestellt werden. In Zeile 120 werden sie mit LSET linksbündig in den Puffer übernommen (RSET = rechtsbündig). Der Einfachheit halber wurde ihnen hier direkt ein String zugewiesen (*Bild 1*). Natürlich kann ihnen auch der Wert einer ande-

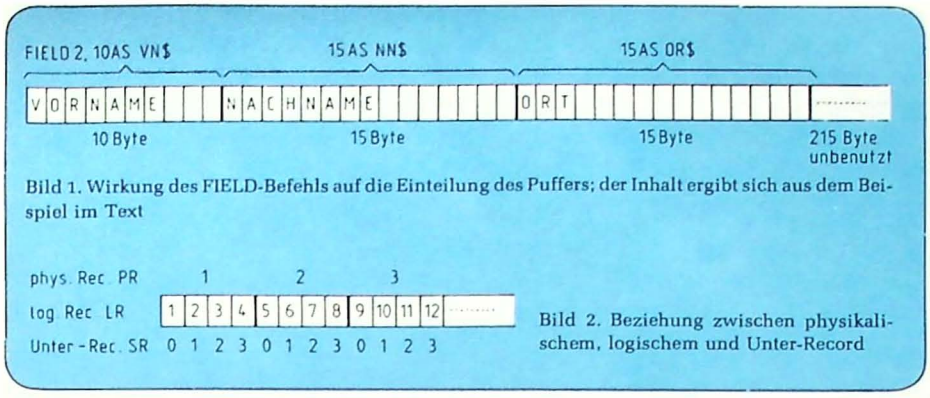


Bild 1. Wirkung des FIELD-Befehls auf die Einteilung des Puffers; der Inhalt ergibt sich aus dem Beispiel im Text

Bild 2. Beziehung zwischen physikalischem, logischem und Unter-Record

ren Variablen zugewiesen werden, der z. B. vorher mit Hilfe von INPUT oder LINE INPUT eingetippt wurde. An dieser Stelle ist zu vermerken, daß alle Variablen als Strings abgespeichert werden. Numerische Variablen muß man deshalb vorher konvertieren. Für die drei möglichen Typen "Integer", "Single Precision" und "Double Precision" sind dazu die Funktionen MKI$, MKS$ und MKD$ vorgesehen. Im Puffer müssen für sie 2, 4 und 8 Byte reserviert werden. Wollte man also im vorhergehenden Beispiel die Variablen A%, B und C# abspeichern, müßte man die Zeilen 110 und 120 ersetzen durch:

```
110 FIELD 2, 2 AS A$, 4 AS B$, 8 AS C$
120 LSET A$ = MKI$(A%) : LSET B$ =
MKS$(B) : LSET C$ = MKD$ (C#)
```

Zeile 130 schließlich bringt den Pufferinhalt auf die Diskette. Wird die Record-Nummer im PUT-Befehl nicht angegeben, nimmt der Computer Eins an und inkrementiert dann bei jeder weiteren Schreiboperation.

Um einen Record zu lesen, benutzt man dieselbe FIELD-Zeile wie beim Schreiben. Danach holt man ihn mit GET File-Nr., Record-Nr. in den Puffer. Die Variablen haben jetzt die abgespeicherten Werte. Numerische Variablen werden mit CVI, CVS und CVD in ihr ursprüngliches Format konvertiert.
Beispiel: A% = CVI (A$)
Der im ersten Beispiel abgespeicherte Record kann mit diesem Programm gelesen werden:

```
200 OPEN"R", 2, "ADRESS"
210 FIELD 2, 10 AS VN$, 15 AS NN$, 15
    AS OR$
220 GET 2, 65
```

Innerhalb einer Datei ist es zulässig, verschiedenen Records verschiedene Feldformate zuzuweisen. Auch beim Lesen muß nicht unbedingt dasselbe Format gewählt werden wie beim Schreiben. Es ist z. B. denkbar, die getrennt abgespeicherten Vor- und Nachnamen als eine Variable wieder von der Diskette zu holen.

Will man zu einer Datei Records lückenlos anfügen, leistet die LOF-Funktion wertvolle Dienste: LOF (File-Nr.) liefert die Nummer des letzten Records.

Was tun gegen Platzverschwendung?

In vielen Fällen wird man abzuspeichernde Daten in kleinere Portionen als 255 Byte einteilen. Wertvoller Speicherplatz bleibt dann ungenutzt. Um dies zu verhindern, empfiehlt das Radio-Shack-Handbuch [1] ein Verfahren, das folgendermaßen aussieht:

Die physikalischen Records werden vom Programm in je vier Unter-Records (*Subrecord*) eingeteilt, die mit Subrecord-Nummern (SR = 0...3) versehen werden. Die neu entstandenen logischen Records beinhalten maximal 63 Byte; sie werden mit logischen Record-Nummern (LR) identifiziert, die mit 1 beginnen. *Bild 2* zeigt das Schema. Bei bekannten LR können PR und SR berechnet werden:

a) PR = INT ((LR-1)/4)+1
b) SR = LR-4* INT ((LR-1)/4)–1

Diese beiden Werte sind nötig, um genau die Portion in den oder aus dem Puffer zu bringen, die dem logischen Record entspricht. Nehmen wir an, N$ sei die interessierende Variable; D$ ist eine Hilfs-Variable. Mit

```
FIELD 3, SR* 63 AS D$, 63 AS N$
```

wird N$ einem von vier Unter-Record-Bereichen zugewiesen. Der physikalische Record, der nach wie vor im PUT- und im GET-Befehl eingesetzt wird, kann wie angegeben errechnet werden. Natürlich ist es auch möglich, N$ durch mehrere andere Variable zu ersetzen. Zusammen dürfen sie 63 Byte belegen – sie müssen es aber nicht. Ebenso steht es einem frei, nicht vier, sondern eine andere Zahl von Unter-Records zu bilden. Eine intensive Verwendung dieser Unter-Record-Technik demonstriert ein „Datenbank"-Programm, das ebenfalls in diesem Sonderheft zu finden ist.

Literatur

[1] Disk Basic Version 1.1, TRS DOS Version 2.2. Handbuch der Fa. Radio Shack, Juli 1978. ■

TRS-80 — ein Computer mit kleinem Fehler

Unbekannte Fehler in der ROM- oder RAM-Software eines Computers führen zu unerkannten oder unerklärlichen Fehlern im Ergebnis. Oft bleiben diese Fehler selbst dem Hersteller verborgen. Hier ein Beispiel: Dem Mikrocomputer TRS-80 von Tandy wird nach dem Einschalten folgendes BASIC-Programm eingegeben.

```
10 FOR I = 0 TO 3
20 READ A
30 PRINT A
40 NEXT I
50 DATA 1, 2, 3, 4
```

Die Analyse dieses einfachen Programmes ergibt, daß der Computer beim Ausführen die Ziffern 1 2 3 4 auf dem Bildschirm anzeigen wird. Tatsächlich erscheint aber viermal die Ziffer 1.

Das dem TRS-80 User's Manual (Level I) entnommene obige Programmbeispiel wird also falsch ausgeführt. Beim ordnungsgemäßen Arbeiten des Programms müßte, durch einen internen Zähler gesteuert, für I = 0 das 1. Element der Datenzeile der Variablen A zugeordnet werden, für I = 1 das 2. Element usw. Das sollte zum Ausdruck der Ziffern 1 2 3 4 führen.

Beim TRS-80 funktioniert dieser Zähler nicht. Das ist manchmal ärgerlich, wenn auf Grund dieser Tatsache einige korrekte, aber umfangreichere BASIC-Programme ähnlichen Typs falsch arbeiten. Der hier beschriebene Fehler erledigt sich aber oft unbemerkt von selbst, wenn der Computer zuvor ein längeres Programm ohne READ-Anweisungen ausführt.

Was aber ist zu tun, wenn sich ein Programm mit READ-Anweisungen im Arbeitsspeicher befindet und der interne Zähler nicht arbeiten will? Einen Ausweg bietet das Kommando:

```
POKE 16553,255
```

Nach einmaliger Eingabe dieses Befehles arbeitet der interne Zähler ordnungsgemäß. *Wolfgang Neske*

Eine der Standardaufgaben für Computer besteht darin, Daten abzuspeichern und sie dem Benutzer bei Bedarf wieder zugänglich zu machen. Das ideale Speichermedium für private Zwecke ist in diesem Fall die Mini-Diskette: Sie nimmt einerseits genügend Information auf (rund 40 Schreibmaschinenseiten) und ist andererseits hinreichend schnell zu durchsuchen. Mit dem vorgestellten Programm, das in BASIC geschrieben wurde, sind TRS-80-Besitzer mit Floppy-Disk-Laufwerk in der Lage, sich eine kleine Datenbank anzulegen.

Rudolf Hofer

Datenbank-Programm für den TRS-80

Betriebsarten

Das Programm kennt fünf Betriebsarten, von denen im Normalfall zwei benutzt werden. In der einen – „Fortlaufend schreiben" – kann der Benutzer Records bis zu einer Länge von 248 Zeichen (auch alle Sonderzeichen sind erlaubt) in die Datei einschreiben. Als Record bezeichnet man ein abgeschlossenes „Datenpaket" – etwa eine Adresse –, das vom Programm als Ganzes behandelt wird. Wo der Record abgelegt wird, braucht den Benutzer nicht zu kümmern – mit einer Ausnahme: wenn die Datei zum ersten Mal mit Daten beschrieben wird.

Die zweite Haupt-Betriebsart nennt sich „Suchen". Sie erfragt vom Benutzer zwei Begriffe und durchsucht dann alle abgespeicherten Records, ob einer der beiden vorkommt oder ob beide vorkommen. Welches Kriterium gewählt wird, bestimmt wiederum der Benutzer. Die gefundenen Records werden ausgedruckt. Als Begriff ist jede beliebige Zeichenfolge bis zu einer Länge von 255 Zeichen erlaubt.

Mit zwei weiteren Betriebsarten können logische Records, deren Bedeutung noch näher erklärt wird, gezielt gelesen und geschrieben werden. In der Praxis verwendet man sie zum Ändern von Records.

In der fünften Betriebsart schließlich teilt man dem Computer mit, wo er bei „fortlaufendem Einschreiben" den nächsten Record ablegen soll. Wie erwähnt, ist das nur ein einziges Mal nötig. Der Computer merkt sich diesen Wert, auch wenn er ausgeschaltet wird, und bringt ihn laufend auf den neuesten Stand.

Bedienung

Wird das Programm zum ersten Mal in Betrieb genommen, startet man es mit RUN 790. Auf dem Bildschirm erscheint dann die Frage LETZTER RECORD? Zweckmäßigerweise gibt man die Zahl 1 ein (nicht vergessen: nach jeder Eingabe Enter-Taste drücken). Die Daten werden dann ab dem logischen Record 2 eingeschrieben. Im logischen Record 1 könnte man den Namen der Datei ablegen, falls es nicht die einzige ist.

Von jetzt ab wird das Programm nur noch mit RUN gestartet. Der Computer löscht daraufhin den Bildschirm und druckt folgenden Text aus:
1 = SCHREIBEN 2 = LESEN 3 = FORTLAUFEND SCHREIBEN 4 = SUCHEN?

Er erwartet dann eine der Zahlen 1...4, mit denen der Benutzer die entsprechende Betriebsart wählt. Bevor die verschiedenen Möglichkeiten näher besprochen werden, soll der Begriff „logischer Record" kurz erklärt werden:

Normalerweise ist die kleinste Datenmenge, die als Record von und zur Diskette transportiert werden kann, 256 Byte. Um nicht zu viel Platz zu verschenken, werden per Programm kleinere „Portionen" zu je 62 Byte benutzt. Sie werden der Reihe nach durchnumeriert und als logische Records bezeichnet.

Fortlaufend schreiben

Der Computer druckt die Nummer des logischen Records aus, der als nächster eingeschrieben wird, und fordert dann den Benutzer auf: TEXT EINGEBEN. Gibt man lediglich eine Null ein, kehrt das Programm zum Anfang zurück, und man kann eine neue Betriebsart wählen. Ein abzuspeichernder Text wird mit der Enter-Taste abgeschlossen, worauf der nächste Record angefordert wird. Vorher kann innerhalb einer Zeile korrigiert werden. Für Wagenrücklauf/Zeilenvorschub wird die Taste „↓" verwendet.

Das Programm ist so aufgebaut, daß vorzugsweise kurze Records mit weniger als 62 Zeichen abgespeichert werden. Längere müssen beim Suchlauf erst miteinander verbunden werden, und das kostet sehr viel Zeit. Trotzdem darf der Benutzer Texte bis zu 248 Zeichen eingeben. Sind sie jedoch im allgemeinen länger als 62 Byte, dann empfiehlt sich eine kleine Programmkorrektur, die einem logischen Record mehr Platz zur Verfügung stellt.

Suchen

Der Computer druckt die Frage BEGRIFFE? aus und erwartet zunächst den ersten Suchbegriff. Mit einer Null kehrt man wieder zum Programmanfang zurück und kann eine neue Betriebsart wählen. Wird ein Suchbegriff eingegeben, so schließt man ihn mit Enter ab und tippt den zweiten ein. Mit einer Eins sagt man dem Computer, es sollen beide Begriffe im Record vorkommen (UND-Verknüpfung); eine Zwei bedeutet, es genügt, wenn ein Begriff vorkommt (ODER-Verknüpfung).

Soll nur nach einem Begriff gesucht werden, kommt der Computer am schnellsten zum Ziel, wenn man dieses Wort als ersten Suchbegriff verwendet, beim zweiten lediglich Enter drückt und sich dann für UND entscheidet. Sind die Daten gefunden, fragt der Computer nach weiteren Begriffen.

Gezielt lesen

Auf die Frage LOG. REC.-NR.? tippt der Benutzer eine Zahl ein, und der entsprechende logische Record wird gelesen. Diese Betriebsart wird man vor allem dann verwenden, wenn man anschließend einen Record ändern will. Im Lese-Modus ist man so lange, bis eine Null eingegeben wird.

Gezielt schreiben

Auch hier gibt man zuerst die Nummer des logischen Records ein. Ist sie Null, kehrt man zum Programmanfang zurück und kann eine neue Betriebsart wählen. Andernfalls fordert der Computer auf: TEXT EINGEBEN. Achtung: Jetzt kann tatsächlich nur ein logischer Record eingegeben werden, d. h. maximal 62 Zeichen. Nachdem man die Eingabe mit der Enter-Taste abgeschlossen hat, erscheint die Frage TEXT BEENDET = ENTER? auf dem Bildschirm.

Bei erneutem Enter kann ein weiterer logischer Record geschrieben werden. Hat der Platz von 62 Zeichen aber nicht ausgereicht, um die gewünschte Information einzugeben, dann drückt man an dieser Stelle nicht die Enter-Taste, sondern tippt die niederwertige Stelle der logischen Record-Nummer ein. Diesselbe Zahl wird bei den unmittelbar folgenden Records eingegeben, sofern sie inhaltlich zum vorhergehenden Text gehören. Maximal dürfen vier logische Records auf diese Weise verknüpft werden.

Ein Beispiel soll diesen Vorgang verdeutlichen: Nehmen wir an, wir hätten festgestellt, die Adresse in den logischen Records 32 und 33 hat sich geändert. Wir wählen die Betriebsart „Gezielt schreiben" und geben die logische Record-Nummer 32 ein. Jetzt verbessern wir die ersten 62 Zeichen der Adresse und drücken die Enter-Taste. Auf dem Bildschirm erscheint TEXT BEENDET = ENTER? Da sich die Änderung auch noch auf den logischen Record 33 bezieht, geben wir jetzt 2 (= niederwertige Stelle der log. Record-Nummer) ein und drücken dann erst Enter. Wieder fragt der Computer nach der log. Record-Nummer. Wir geben 33 ein und setzen dann unsere Korrektur fort. Nach Enter erscheint wieder TEXT BEENDET = ENTER? Jetzt wird noch einmal 2 eingetippt und mit Enter abgeschlossen. Nun weiß der Computer, daß die logischen Records 32 und 33 zusammengehören. Um in eine neue Betriebsart zu gelangen, beantworten wir die neuerliche Frage nach der log. Record-Nummer mit Null.

Programmdetails

Die vier Hauptprogrammteile bestehen aus Schleifen, die durch den ON-Befehl in Zeile 130 wahlweise angesprungen werden und die man mit einer Null als Antwort auf die jeweils erste Frage wieder verlassen kann (*Bilder 1* und *2*). Wesentlicher Bestandteil ist das Unterprogramm für die Feldvereinbarung ab Zeile 2000, das für die richtige Plazierung der logischen Records innerhalb der physikalischen Records sorgt. Es berechnet bei bekanntem LR

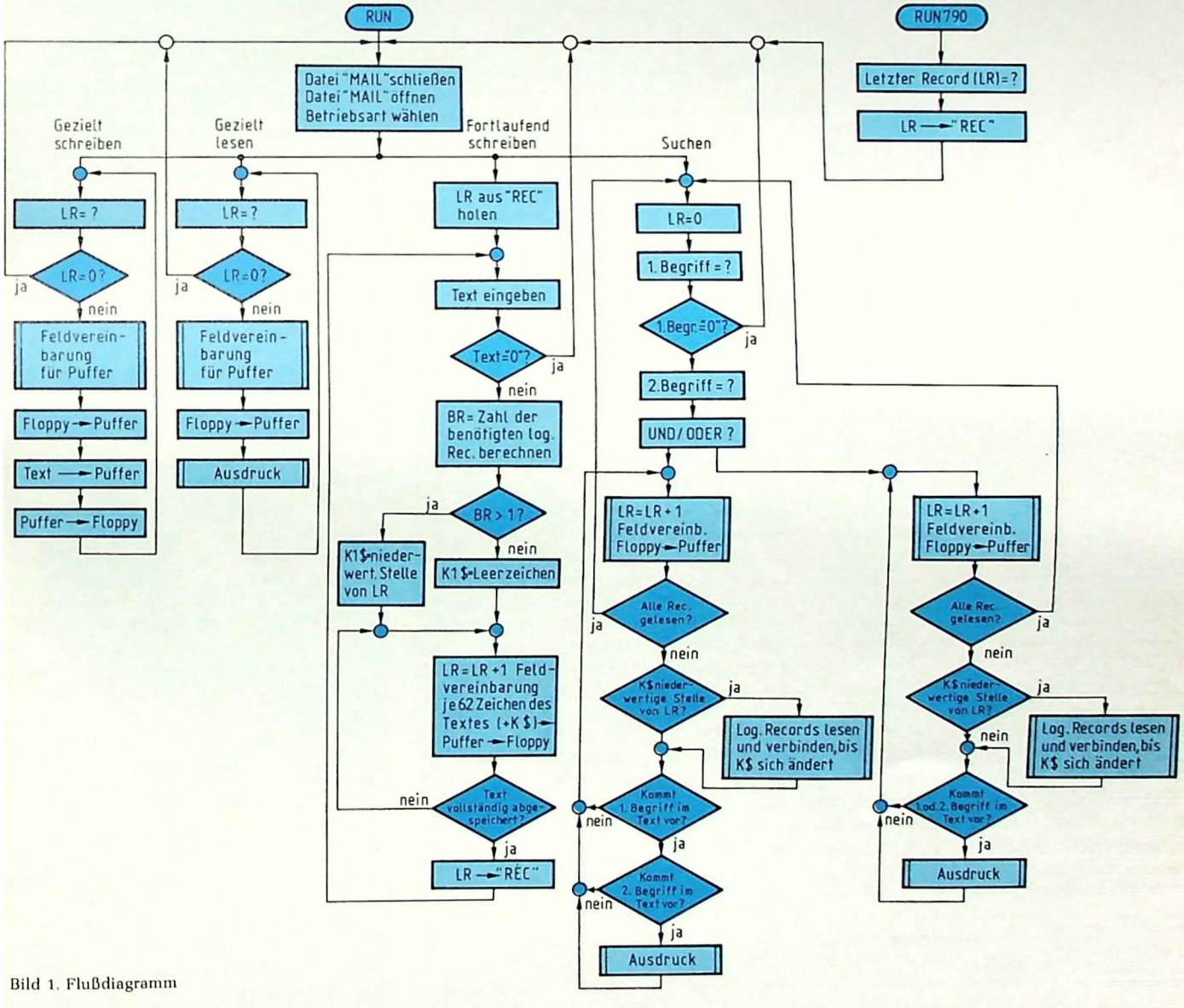


Bild 1. Flußdiagramm

```
99 REM KOMMANDOSCHLEIFE
100 CLEAR1000:CLS:CLOSE
110 PRINT:INPUT"1=
SCHREIBEN 2=LESEN
3=FORTLAUFEND SCHREIBEN
4=SUCHEN";N
120 OPEN"R",1,"MAIL"
130 CLS:ON N GOTO200
,300,400,560
199 REM GEZIELT
SCHREIBEN
200 PRINT:INPUT"LOG.
RECORD-NR.";LR
210 IF LR=0 THEN100
220 GOSUB2000
230 GET1,PR
235 PRINT"TEXT EINGEBEN"
240 LINE INPUT "";A$:LSET
N$=A$
250 INPUT"TEXT
BEENDET=ENTER";A$:LSET
K$=A$
280 PUT 1,PR:GOTO200
299 REM GEZIELT LESEN
300 PRINT:INPUT"LOG.
RECORD-NR.";LR
310 IF LR=0 THEN 100
320 GOSUB2000
330 GET 1,PR
340 GOSUB3100
350 GOTO300
399 REM FORTLAUFEND
SCHREIBEN
400 OPEN"I",2,"REC"
410 INPUT#2,LR
420 CLOSE 2
475 PRINT"LOG.
REC.-NR.="; LR+1:PRINT:
PRINT"TEXT
EINGEBEN":PRINT
480 LINE INPUT"";
A$:Z=LEN(A$)
481 IF A$="0" THEN 100
482 BR=INT(Z/62+0.99):
N=0
483 IF BR>1 THEN
K1$=RIGHT$(STR$(LR+1),1)
ELSE K1$=" "
484 GOSUB3000
485
A1$=MID$(A$,N%62+1,62):
LSET N$=A1$:LSET K$=K1$
488 N=N+1
490 PUT 1,PR
492 IF N<BR THEN 484
493 OPEN"O",2,"REC"
494 PRINT#2,LR
496 CLOSE 2
499 GOSUB 2000
530 GOTO 475
559 REM SUCHEN
560 INPUT"BEGRIFFE";C$
565 LR=0
570 IF C$="0" THEN100
575 INPUT F$
580 INPUT"UND=1
ODER=2";N
590 ON N GOTO 600,700
600 GOSUB 3000
615 IF LOF(1)%4<LR
THEN560
617 IFK$=RIGHT$(STR$(LR)
,1) GOSUB 3300
620 IF INSTR(N$,C$)=0
THEN 600
630 IF INSTR(N$,F$)=0
THEN 600
640 GOSUB 3100
650 GOTO600
700 GOSUB 3000
715 IF LOF(1)%4<LR
THEN560
717 IFK$=RIGHT$(STR$(LR)
,1) GOSUB 3300
720 IF INSTR(N$,C$)=0 AND
INSTR(N$,F$)=0THEN 700
730 GOSUB 3100
740 GOTO 700
769 REM LETZTEN RECORD
FESTLEGEN
790 INPUT"LETZTER
RECORD";LR
800 OPEN"O",2,"REC"
810 PRINT#2,LR
820 CLOSE2
830 GOTO100
1999 REM UNTERPROGRAMM
FUER FELDVEREINBARUNG
2000 SR=LR-4*INT((LR-1)
/4)-1
2020 FIELD 1,SR%63 AS
D$,62 AS N$,1 AS K$
2030 PR=INT((LR-1)/4)+1
2040 RETURN
2999 REM LESEN
3000 LR=LR+1
3010 GOSUB2000
3020 GET 1,PR
3030 RETURN
3099 REM AUSDRUCKEN
3100 CLS:PRINT:PRINT
3105 PRINT"LOG.
REC.-NR=";LR
3107 PRINT:PRINT
3110 PRINT N$
3130 PRINT:RETURN
3299 REM VERBINDEN
3300 K1$=K$:M$=CHR$(0)
3310 M$=M$+N$
3320 GOSUB 3000
3330 IF K$=K1$ THEN 3310
3340 N$=M$
3350 LR=LR-1:RETURN
```

Bild 2. Das Listing wurde mit Hilfe eines Formatier-Programmes ausgegeben, das die Zeilenlänge beschränkt. Jede Zeile, die nicht mit einer Zahl beginnt, ist deshalb an die vorhergehende anzuschließen. Ob ein Leerzeichen eingefügt wird, ist im allgemeinen ohne Bedeutung

die Werte PR und SR, wie es in [1] beschrieben ist. Damit braucht man im übrigen Programm nur noch mit logischen Record-Nummern (LR) zu arbeiten.

Außer der eigentlichen (wahlfreien) Datei „MAIL" wird eine zweite (sequentielle) verwendet, die mit „REC" bezeichnet wird. Sie enthält die Nummer des letzten beschriebenen logischen Records und wird bei „fortlaufendem Schreiben" automatisch inkrementiert.

Um sich das Prinzip klarzumachen, betrachtet man sich am besten die Programmteile „Gezielt schreiben" (200...280) und „Gezielt lesen" (300...350), die geradlinig und ohne Hinterhältigkeiten ablaufen. Erweiterungen davon sind die Betriebsarten „Fortlaufend schreiben" und „Suchen".

Betrachten wir uns erstere für den Fall, daß ein Text mit weniger als 63 Zeichen abzuspeichern ist. Die Zahl der benötigten logischen Records BR ist dann 1. K1$ ist eine Hilfs-Variable, deren Wert später K$ zugewiesen wird. K$ aber ist ein Zeichen, das mit jedem logischen Record abgespeichert wird und Auskunft darüber gibt, ob der nächste logische Record mit dem vorhergehenden zu verbinden ist. Bei Texten mit weniger als 63 Zeichen erhält K$ das Leerzeichen zugewiesen. Der Text ist also im betrachteten Fall nach dem ersten Durchlauf vollständig abgespeichert, und weitere Daten können eingegeben werden.

Hat der Text mehr als 62 Zeichen, wird er in mehreren logischen Records abgespeichert, und K$ erhält jeweils den Wert der niederwertigen Stelle der ersten logischen Record-Nummer (*Bild 3*).

Beim Suchen werden zuerst die beiden Begriffe eingegeben, dann verzweigt das Programm, abhängig davon, ob man sich für UND oder ODER entschieden hat. Diese beiden Teile unterscheiden sich allein dadurch, daß bei ODER jedesmal beide Begriffe in einem Record gesucht werden müssen, während bei UND zuerst nach einem gesucht wird und erst bei Erfolg nach dem zweiten. Das wirkt sich auf die Geschwindigkeit aus, weshalb man immer UND wählen sollte, wenn man nur ein Suchwort hat (siehe Bedienung).

Da man beim Einschreiben die Möglichkeit hat, den Text einzugeben, ohne auf die Record-Grenzen zu achten, kann es vorkommen, daß man ein Wort in der Mitte trennt. Damit auch dieses Wort als Ganzes wieder gefunden wird, muß man zusammengehörende Records verbinden, bevor nach einer Zeichenfolge gesucht wird. Dies besorgt das Unterprogramm ab Zeile 3300, das abhängig vom Wert der Variablen K$ angesprungen wird (*Bild 3*). Da diese zusätzlichen Befehle sehr zeitaufwendig sind, sollte man möglichst darauf verzichten, sehr oft längere Records zu verwenden.

Wer im allgemeinen längere Texte abspeichern will, kann folgende Änderungen vornehmen, die das Programm für einen durchschnittlichen Bedarf von knapp 126 Zeichen optimieren:

Zeile 482: 62 durch 126 ersetzen, 0.99 durch 0.999 ersetzen
Zeile 486: 62 zweimal ersetzen durch 126
Zeilen 615, 715 und 2000: Jede 4 ersetzen durch 2
Zeile 2020: 63 ersetzen durch 127, 62 ersetzen durch 126
Zeile 2030: 4 ersetzen durch 2

Der Zeitbedarf für das Suchen kann nur grob angegeben werden. Er beträgt etwa 200 ms/log. Record, wenn die empfohlene Länge nicht überschritten wird (bei Überschreiten etwa das Vierfache). Das Programm belegt insgesamt 1600 Byte RAM.

LR	Text (N$)	K$
11	Text 1	
12	Text 2	
13	Text 3	3
14	Text 3	3
15	Text 3	3
:	:	
23	Text 7	
24	Text 8	4
25	Text 8	4
26	Text 9	
27	Text 10	7
28	Text 10	7

Bild 3. Zusammengehörige Texte werden durch den Wert von K$ ausgedrückt: Falls ein logischer Record (LR) für sich steht, wird ein Leerzeichen abgespeichert, andernfalls erhält K$ die niederwertige Stelle des ersten LR zugewiesen. Dies ist willkürlich festgelegt; wichtig ist allein, daß man ein Kriterium erhält, mit dem man beim Suchen die Records wieder richtig verbinden kann

Superhirn

Das in Basic geschriebene Superhirn-Programm ermöglicht es, ohne Spielpartner zu spielen, und belegt einen Speicherplatz von 1563 Byte. Nach dem Start stellt es sich selbst vor: Es werden die Spielregeln ausgegeben. Anschließend erfolgt die Eingabe für die gewünschte Ziffernlänge. Hierbei kann zwischen einer vier- und fünfstelligen Zahl gewählt werden. Es folgt die Ausgabe des Spielfeldes.

In den Zeilen 13...16 werden die einzelnen Ziffern der Zufallszahl ermittelt. Hierbei wird die Zeit „TI" als Zufallsgröße in den Befehl „RND" eingebracht. Die Zeile 16 stellt sicher, daß die Ziffern immer im Bereich von 1...9 liegen. Nun folgt ein Sprung zur Zeile 1000. Hier wird sichergestellt, daß keine Ziffer mehrfach gespeichert wird. Mit dem Befehl „ON X GOTO" werden die einzelnen Ziffern in die Variablen AA...EE abgespeichert. Anschließend erfolgt der Rücksprung zur Zeile 20. Hier wird verglichen, ob alle Ziffern abgespeichert sind. Ist dies der Fall, so kann die Eingabe eines Rateversuches erfolgen. Dies geschieht in Zeile 25 in der Form eines Strings. Zeile 26 stellt den Versuchszähler dar. Die Zeilen 27 oder 28 bewirken einen Abbruch des Spiels mit der Ausgabe der zu ermittelnden Zahl.

In den Zeilen 31...35 wird der String in einzelne Zahlvariablen umgewandelt. Ab Zeile 500 werden der Versuch ausgewertet und die jeweiligen Zähler betätigt. In dem Moment, in dem der Zähler „J" mit der Variablen „K" identisch ist, wird das Spiel beendet und das Ergebnis ausgegeben. Wenn der Vergleich „J – K" negativ ausfällt, werden die Variablen „V,J,N" ausgegeben und „J" und „N" auf Null gesetzt. Es erfolgt ein Sprung zur Zeile 25, wo eine neue Eingabe erfolgt. Die Zeilen 2002...2004 geben Auskunft über das Endergebnis. Die Zeilen 2004...2006 ermöglichen ein neues Spiel, ohne das Programm neu starten zu müssen.

Die Eingabe erfolgt als String und wird in vier oder fünf Zahlvariablen umgewandelt. Dieser Weg wurde beschritten, um den PET 2001 nicht in die Verlegenheit zu bringen, eine vier- oder fünfstellige Zahl mittels mathematischer Befehle in einzelne Ziffern umwandeln zu müssen, da er bekanntlich nicht der genaueste Rechner im Nachkommabereich ist.

Leider kommt es leicht vor, daß sich bei der manuellen Programmeingabe Fehler einschleichen. Um nun das Programm überprüfen zu können, kann man folgende Zeile einschreiben:

21 PRINT AA;BB;CC;DD;EE

Hier wird nun das richtige Ergebnis sofort eingeblendet. Bei einer vierstelligen Zahl wird die fünfte Stelle als Null ausgegeben; sie bleibt dann unberücksichtigt.

Walter Dochow

Verzeichnis der Variablen

Variable	Funktion
K	legt die Ziffernlänge fest
A	enthält die einzelnen Ziffern der Zufallszahl
X	dient zum berechneten Sprung
AA...EE	enthalten die Zufallszahl
A$	Eingangsvariable
V	Zähler für die Anzahl der Versuche
A1...A5	enthalten die Ziffern der Eingabe
J	Zähler für die Anzahl der richtigen Ziffern am richtigen Platz
N	Zähler für die Anzahl der richtigen Ziffern am falschen Platz

```
1 PRINT"■"
2 PRINT"R S U P E R H I R N ■"
3 PRINT:PRINT"DU MUST EINE VIER- ODER FUENFSTELLIGE   ZAHL ERRATEN.";
4 PRINT" JEDE ZIFFER KOMMT NUR EIN-MAL VOR."
5 PRINT"SOLLTEST DU DIE ZAHL NICHT ERRATEN, SO  KANNST DU SIE DURCH",
6 PRINT"EINGABE DES ZEICHENSR @ ■ ABRUFEN."
10 CLR:PRINT"SOLL DIE ZAHL R 4 ■ ODER R 5 ■ ZIFFERN HABEN" :INPUTK:PRINT"■"
11 PRINT"     RVERSUCH   NR.  PLATZ    ZAHL■"
13 FORX=1TOK
15 A=INT(10*(RND(TI)))
16 IFA<00RA>9THEN15
19 GOTO1000
20 NEXT
25 PRINT:INPUT"     ";A$
26 V=(V+1)
27 IFK=4ANDA$="@"THENPRINT"R",AA;BB;CC;DD"■":PRINT:GOTO2004
28 IFK=5ANDA$="@"THENPRINT"R",AA;BB;CC;DD;EE"■":PRINT:GOTO2004
31 AA$=LEFT$(A$,1):BB$=MID$(A$,2,1)
32 CC$=MID$(A$,3,1):DD$=MID$(A$,4,1)
33 EE$=RIGHT$(A$,1):A5=VAL(EE$)
34 A1=VAL(AA$):A2=VAL(BB$)
35 A3=VAL(CC$):A4=VAL(DD$)
40 GOTO500
500 IFA1=AATHENGOSUB600
501 IFA1=BBORA1=CCORA1=DDTHENGOSUB700
502 IFA2=BBTHENGOSUB600
503 IFA2=AAORA2=CCORA2=DDTHENGOSUB700
504 IFA3=CCTHENGOSUB600
505 IFA3=AAORA3=BBORA3=DDTHENGOSUB700
506 IFA4=DDTHENGOSUB600
507 IFA4=AAORA4=BBORA4=CCTHENGOSUB700
508 IFK=4THEN550
509 IFA5=EETHENGOSUB600
510 IFA1=EEORA2=EEORA3=EEORA4=EETHENGOSUB700
511 IFA5=AAORA5=BBORA5=CCORA5=DDTHENGOSUB700
550 PRINT"■";TAB(15);V;TAB(21);J;TAB(28);N;TAB(32)
551 J=0:N=0
552 GOTO25
600 J=(J+1)
601 IFJ=KTHENGOTO2000
603 RETURN
700 N=(N+1)
701 RETURN
1000 IFA=AAORA=BBORA=CCORA=DDTHEN15
1004 IFK=4THEN1010
1005 IFA=EETHEN15
1010 ONXGOTO1100,1200,1300,1400,1500
1100 AA=A:GOTO20
1200 BB=A:GOTO20
1300 CC=A:GOTO20
1400 DD=A:GOTO20
1500 EE=A:GOTO20
2000 PRINT:PRINT"■■";TAB(15);V;TAB(21);J;TAB(28);N;TAB(32)
2002 PRINT:PRINT:PRINT"HERZLICHEN GLUECKWUNSCH, DU HAST RICHTIGGERATEN !!!"
2004 PRINT:PRINT:PRINT"WUENSCHST DU EIN NEUES SPIEL? [J/N ":INPUTY$
2006 IFY$="J"THEN1
2008 PRINT"■":PRINT"VIELEN DANK FUER DAS SPIEL. AUF WIEDER- SEHEN.":END
```

Basic-Listing des Superhirn-Programms. Die unterstrichenen Zeichen werden auf dem PET 2001 invers dargestellt

Kalender-Algorithmus

In dem Beitrag in Heft 22/1978, S. 1102, ist bei der Angabe der Schaltregel für den Gregorianischen Kalender ein Fehler unterlaufen. Daß alle durch 4 teilbaren Jahre Schaltjahre sind, ist allgemein bekannt. Weniger geläufig ist jedoch die durch die Gregorianische Kalenderreform 1582 eingeführte Ausnahme, daß von den Jahrhundertjahren nur diejenigen Schaltjahre sein sollen, die durch 400 teilbar sind. 1600, 2000, 2400 sind also Schaltjahre, 1700, 1800, 1900, 2100, 2200 sind keine Schaltjahre.

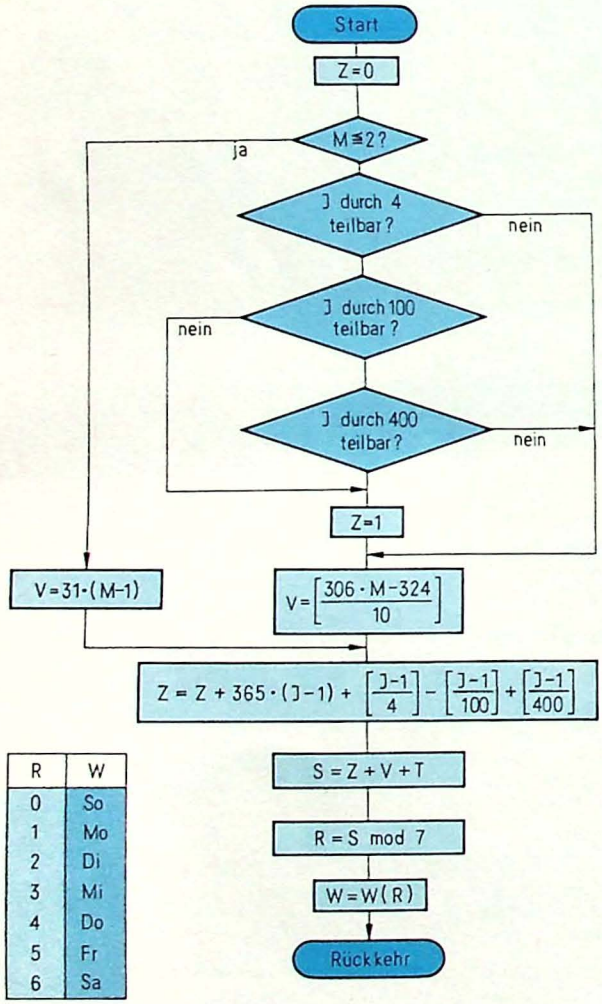
Ziel der Reform war es, das aus ganzen Tagen bestehende Kalenderjahr dem nicht in ganzen Tagen meßbaren Lauf der Erde um die Sonne besser anzupassen, so daß z. B. der Frühlingsanfang am 21. März erhalten bleibt.

Entsprechend dem Fehler in der Schaltregel ist in Heft 22 auch der Gültigkeitsbereich für das Programm falsch angegeben: Nicht vom 1. 3. 1600 bis 28. 2. 2000, sondern vom 1. 3. 1900 bis 28. 2. 2100 ist das Programm anwendbar.

Die Korrekturen seien zum Anlaß genommen, einen Kalender-Algorithmus vorzustellen, der keinen eingeschränkten Gültigkeitsbereich hat (schon in Heft 22 wurde erwähnt, daß dafür ein nur geringer Mehraufwand nötig ist) und der für die Ermittlung der bis zum Vormonat des Monats M vergangenen Tage V keine Tabelle, sondern eine für Mikrocomputer geeignete Formel benutzt (*Bild 1*). Im folgenden soll nicht das gesamte Biorhythmus-Programm wiederholt werden, sondern nur eine Alternative zu dem separat verwendbaren Kalender-Unterprogramm angegeben werden (*Bild 2*). Damit das Programm auch ohne Kenntnis von Heft 22 verständlich wird, seien zuvor noch einige Abkürzungen und Symbole erläutert. Zu den gegebenen Werten Tag T, Monat M, Jahr J (vierstellig) eines Datums im Gregorianischen Kalender werden die fortlaufende Tagesnummer S und der Wochentag W berechnet. Die fortlaufende Tagesnummer ist gleichzeitig die Anzahl der von einem Bezugspunkt Null bis zum gegebenen Datum vergangenen Tage. Sie ist ein Hilfsmittel für die Ermittlung der exakten Differenz (in Tagen) zwischen zwei gegebenen Daten (z. B. Geburtstag und heutiger Tag).

Zum Flußdiagramm sei noch vermerkt, daß die eckigen Klammern angeben, daß nur der ganze Teil des Ergebnisses zu nehmen ist, Stellen nach dem Komma bzw. Bruchteile also wegzulassen sind. Und „mod 7" gibt an, daß nur der bei Division durch 7 sich ergebende Rest zu nehmen ist. Das BASIC-Programm ist an die Form von Heft 22 angelehnt, damit interessierten Lesern der Vergleich erleichtert wird.

Jindra Singer



R	W
0	So
1	Mo
2	Di
3	Mi
4	Do
5	Fr
6	Sa

◄ Bild 1. Flußdiagramm des Kalender-Algorithmus

```
Ø11Ø INPUT T,M,J
Ø12Ø Z=Ø
Ø122 IF M<=2 THEN 296
Ø13Ø IF J/4<>INT(J/4) THEN 292
Ø132 IF J/1ØØ<>INT(J/1ØØ) THEN 29Ø
Ø134 IF J/4ØØ<>INT(J/4ØØ) THEN 292
Ø29Ø Z=1
Ø292 V=INT((3Ø6*M-324)/1Ø)
Ø294 GO TO 3ØØ
Ø296 V=(M-1)*31
Ø3ØØ Z=Z+(J-1)*365+INT((J-1)/4)
Ø3Ø1 Z=Z-INT((J-1)/1ØØ)+INT((J-1)/4ØØ)
Ø34Ø S=Z+V+T
Ø35Ø R=S-INT(S/7)*7
Ø37Ø FOR I=1TO7
Ø38Ø READ W$(I)
Ø39Ø NEXT I
Ø4ØØ PRINT W$(R+1)
Ø51Ø DATA SONNTAG,MONTAG,DIENSTAG,MITTWOCH,
DONNERSTAG,FREITAG,SAMSTAG
```

Bild 2. Dieses BASIC-Programm berücksichtigt auch die „Ausnahme von der Regel" im Gregorianischen Kalender

apple

Wenn beispielsweise im kaufmännischen Bereich bei großen Abrechnungen oder im Bereich naturwissenschaftlich-technischer Anwendungen bei der Auswertung umfangreicher Meßreihen ausgedehnte Zahlensammlungen in ein Rechnersystem einzugeben sind, dann summiert sich die Tipparbeit leicht zu stundenlanger konzentrierter Tätigkeit. Jeder Handgriff, jede einzelne Tastenbetätigung, die ein geeignetes Programm dem Bediener abnimmt, zahlt sich dann in einer deutlichen Verkürzung der Arbeitszeit und einer eklatanten Schonung von Nervenkraft aus. Das nachstehend geschilderte Eingabeverfahren INGET zeigt eine Möglichkeit dazu auf.

Hans-Georg Joepgen

„INGET" verkürzt Tipp-Arbeit

Die meistverbreiteten Basic-Betriebssysteme für Mikrocomputer enthalten zwei verschiedene Verfahren zur Datenübernahme von der Tastatur in den Rechner: einmal die „INPUT"-Prozedur; hier wartet das Programm, bis der Benutzer eine Zeichenkette getippt und sie durch die Betätigung einer Schlußtaste abgeschlossen hat („End Line" beispielsweise beim HP-85, „Enter" beim TRS-80, „Return" bei Rechnern wie ITT 2020, Apple, PET und PC-100). Die eingegebenen Zeichen werden auf dem Schirm mitgeschrieben und dann in eine Variable eingelesen, die dem INPUT-Statement im Programm folgt.

Die zweite verbreitete Prozedur zur Datenübernahme vom Tastenfeld wird durch das Statement GET, wieder gefolgt vom Namen einer Variablen aufgerufen. Hier nun benehmen sich verschiedene Mikrocomputer unterschiedlich. Rechner wie der PET weisen der Variablen unmittelbar den Tastenfeld-Wert zu, auch dann, wenn keine Taste betätigt ist und als Zeichen nur ein Leerstring zurückkommt: es wird nicht gewartet, bis die Taste gedrückt wurde. Soll gewartet werden, muß dieses durch Software erzwungen werden, etwa durch einen Befehl der Art

10 GET A$: IF A$ = " " THEN 10

Rechner wie der ITT 2020 nehmen ihrem Programmierer diese kleine Zusatzmühe selbsttätig ab und warten, wenn sich zum Zeitpunkt des GET-Aufrufs noch kein Zeichen im Tastenspeicher befindet.

Beiden Rechnerklassen, ob mit und der ohne Warte-Automatik, ist gemeinsam, daß sowohl die INPUT- als auch die GET-Prozedur über betriebliche Vorzüge verfügt, die man sich gern vereint wünscht, die sich aber nicht ohne weiteres vereinigen lassen: Bei INPUT hat es der Bediener in der Hand, wann der Rechner bei der Dateneingabe die Kette als beendet ansieht und seine Arbeit fortsetzt, bei GET reicht die Betätigung einer einzigen Taste, den gewünschten Vorgang auszulösen. Einfach gestalten sich die Verhältnisse, wenn schon beim Schreiben des Programms feststeht, in welcher Reihenfolge manuelle Dateneingaben welcher Art erfolgen: Bei Einzeichen-Angaben wird man GET wählen, ebenso bei Mehrzeichen-Eingaben, deren Zeichenanzahl feststeht, etwa so für eine zwölfstellige Kette:

```
10 S$ = " ": REM SAMMELSTRING LOESCHEN
20 FOR N = 1 TO 12
30 GET UE$ : REM UEBERNAHMESTRING
40 IF UE$ = " " THEN 30 : REM ENTFAELLT BEI RECHNERN MIT WARTEAUTOMATIK
50 S$ = S$ + UE$ : NEXT N
```

In allen anderen Fällen muß man INPUT benutzen, insbesondere dann, wenn offenbleiben muß, ob der Bediener nun eine Einzeichen- oder eine Mehrzeicheneingabe vornehmen wird. Bei 3000 Eingaben bedeutet dies beispielsweise 3000 zusätzliche Betätigungen der Abschlußtaste: Man sieht, es lohnt sich, nach einer dritten Eingabeprozedur zu suchen, einer Mischform aus GET und INPUT, die nur dann nach einem Drücken der Schlußtaste verlangt, wenn dieses unumgänglich ist. Bewährt bei mancherlei länglichen Tipparbeiten hat sich beim Verfasser nun eine INGET genannte Prozedur, die wie folgt gekennzeichnet ist:

1. Das zuerst eingegebene Zeichen wird mit GET abgeholt. Der Rechner prüft, ob dies eine plausible Eingabe für eine Ein-Zeichen-Eingabe ist (beispielsweise ‚S' für Summieren, Summe ausdrucken, Summenspeicher löschen, Neueingabe erwarten – oder aber ‚Z' für Zwischensumme drucken, Summenspeicher unverändert lassen, fortfahren – oder ‚=' für Rechnungsabschluß und Folgeprogramm laden).

2. Ergibt die Plausibilitätsprüfung, daß keine für sich allein gültige Ein-Zeichen-Eingabe erfolgt ist, schaltet der Rechner auf INPUT. Der so abgeholten Kette muß nun jedoch noch das zuvor durch GET abgeholte Zeichen hinzugefügt werden, weiterhin ist das Schirmbild entsprechend zu korrigieren.

```
]LIST
10  REM DEMONSTRATION ZU INGET
20  REM ======================
30  REM
40  REM H.-G.  JOEPGEN (070180)
50  REM
60  HOME : PRINT " ZEITENADDITION
     ": PRINT " --------------"

70 SM = 0:SS = 0
80  PRINT
90  PRINT "WERT? ";: GET A$
100  IF A$ = " " THEN B$ = "": GOSUB
     200: GOTO 70
110  IF A$ = "/" THEN  HTAB 1: FOR
     N = 1 TO 39: PRINT " ";: NEXT
     : PRINT " ": PRINT : PRINT :
      PRINT "   +++  BYE.┘": END

120  IF A$ = "Z" THEN B$ = "ZWISC
     HEN": GOSUB 200: GOTO 80
130  PRINT A$;
140  INPUT "";B$:A$ = A$ + B$
150 A =  VAL (A$):M =  INT (A)
160 S =  INT ((A - M) * 100 + 0.5
     )
170 SM = SM + M:SS = SS + S
180  IF SS > 59 THEN SS = SS - 60
     :SM = SM + 1: GOTO 180
190  GOTO 80

200  HTAB 1: PRINT "   +++   "B$"
     SUMME  "SM":";
210  IF SS < 10 THEN  PRINT "0";
220  PRINT SS: RETURN
```

Bei 1000 Zahleneingaben spart INGET genau 100 Tastendrücke – nämlich das Betätigen von „Return" bzw. „Enter"

3. Von all dem bemerkt der Bediener nur soviel, daß der Rechner keine überflüssigen Betätigungen der Abschlußtaste von ihm verlangt. Der Computer benimmt sich, als verfügte er tatsächlich firmwaremäßig über eine dritte Eingabeprozedur ‚INGET'.

Das Programmlisting „Demonstration zu INGET" zeigt, wie diese dritte Eingabeprozedur auf einer Maschine ITT 2020 implementiert wurde. Durch das Programm werden Zeitangaben im Format Minuten-Punkt-Sekunden unter Verwendung von INGET addiert. Folgende Steuerzeichen erkennt der Rechner und folgt ihnen ohne Betätigung von „Return":

Space – Summe ausdrucken. Summenspeicher löschen.

„Z" – Zwischensumme ausdrucken. Als solche kennzeichnen.

„/" – Schirm aufräumen. Abschiedsmeldungdrucken. Ende.

Ziffern – Als numerische Eingabe deuten und auf Input schalten.

Die Variablen im Demonstrationsprogramm haben folgende Funktion:

SM Summenspeicher für Minuten

SS Summenspeicher für Sekunden

S Sekundenspeicher

M Minutenspeicher

A$ Erstzeichen, String mit einem Element sowie Sammelstring

B$ String mit den Folgezeichen, zugleich als Promptingvariable für Zwischensummenausdruck benutzt

A Numerischer Benutzer-Akkumulator

Das Verfahren zur Korrektur des Schirmbildes, Zeilen 140 und 200, ist maschinenspezifisch und hier für Rechner der Apple-Klasse angegeben. Bei der Verwendung auf dem PET ist die HTAB-Funktion durch entsprechende Cursorpositionierung an den Anfang der aktuellen Zeile zu ersetzen und darüber hinaus noch der Tatsache Rechnung zu tragen, daß der PET bei jeder Input-Operation ein – hier unerwünschtes – Fragezeichen ausdruckt, das noch gesondert überschrieben werden muß. ■

```
0000 A9 00     LDA #00
0002 85 FA     STA FA
0004 A9 02     LDA #02
0006 85 FB     STA FB
0008 A2 05     LDX #05
000A BD D7 1F  LDA 1FD7,X
000D 20 00 01  JSR 0100
0010 CA        DEX
0011 D0 F7     BNE 000A
0013 A4 EE     LDY EE
0015 B1 FA     LDA (FA),Y
0017 C9 21     CMP #21
0019 30 05     BMI 0020
001B 88        DEY
001C D0 F7     BNE 0015
001E A4 EE     LDY EE
0020 84 F3     STY F3
0022 20 63 1F  JSR 1F63
0025 A0 00     LDY #00
0027 B1 FA     LDA (FA),Y
0029 D0 03     BNE 002E
002B 4C 4F 1C  JMP 1C4F
002E C9 0D     CMP #0D
0030 F0 D6     BEQ 0008
0032 20 00 01  JSR 0100
0035 C6 F3     DEC F3
0037 D0 E9     BNE 0022
0039 F0 F5     BEQ 0030
003B 00        BRK
```

Bild 1. Hauptprogramm zum Ausdruck gespeicherter Texte

Textausdruck mit Baudot-Fernschreiber

In Heft 24/1978 wurde für den Mikrocomputer KIM-1 (µP: 6502) unter dem Titel „KIM auf Datensuche" ein Karteiprogramm vorgestellt, das die Speicherung von Adressen, Telefonnummern oder anderen Texten und Daten gestattet. Ein preiswerter gebrauchter Fernschreiber ermöglicht mit dem nachfolgenden Programm den Ausdruck einer solchen Kartei.

Weil jeder einzelne Informationsblock länger als eine Fernschreiber-Zeile sein kann, nämlich bis zu 255 Zeichen, muß das Druckprogramm auch dafür sorgen, daß rechtzeitig am Ende einer Zeile eine neue Zeile begonnen wird. Um dies zu bewerkstelligen, ohne ein Wort zu „zerreißen", wird derjenige Leerraum ermittelt, der sich – je nach gewünschter Zeilenlänge – für die Trennung eignet.

Bild 1 enthält das Hauptprogramm, das ein Unterprogramm zur Ausgabe eines Baudot-Zeichens (*Bild 2*) aufruft. Dieses Unterprogramm verwendet wiederum eine ASCII-Baudot-Umwandlungstabelle, die in *Bild 3* aufgelistet ist. Die beiden letztgenannten Programmteile stimmen weitgehend mit dem Programm aus Heft 1/1979 überein und wurden daher nur in hexadezimaler Form wiedergegeben.

Die gewünschte Zeichenzahl pro Zeile muß in die Zelle 00EE eingegeben werden, und zwar als hexadezimaler Wert; für 62 Zeichen pro Zeile muß 00EE also hex 3E enthalten.

Für diejenigen Leser, die keinen KIM-1 besitzen, sei noch die Bedeutung der Monitor-Adressen erwähnt. An der Adresse 1FD7 befindet sich die Zeichenfolge 00 00 00 00 0A 0D, die zur Ausgabe von Wagenrücklauf und Zeilenvorschub dient. Die Null-Zeichen sollen dem Wagen des Fernschreibers genügend Zeit zum Rücklauf geben. Die Adresse 1C4F ist die Startadresse des Monitorprogramms, zu der bei Beendigung des Druckvorganges gesprungen wird, und bei 1F63 steht ein ROM-Unterprogramm, das den Textzeiger 00FA, 00FB (16-bit-Zeiger) um 1 erhöht.

Herwig Feichtinger

Bild 2. ► Unterprogramm zur Ausgabe eines ASCII-Zeichens auf einem Baudot-Fernschreiber. Er ist über eine Treiberstufe (z. B. Darlington-Transistor) an den KIM-Port PA 7 anzuschließen

```
0100 86 F5 A2 FF 8E 01 17 C9
0108 0D D0 04 A9 08 D0 0E C9
0110 0A D0 04 A9 02 D0 06 29
0118 3F AA BD 70 01 85 F6 29
0120 20 C5 F7 F0 0E 85 F7 A8
0128 F0 04 A9 1B D0 02 A9 1F
0130 20 3D 01 A5 F6 20 3D 01
0138 A0 FF A6 F5 60 A2 00 8E
0140 00 17 20 64 01 A0 05 4A
0148 90 04 A2 FF D0 02 A2 00
0150 8E 00 17 20 64 01 88 D0
0158 EE A2 FF 8E 00 17 A2 1E
0160 20 66 01 60 A2 14 8E 07
0168 17 2C 07 17 10 FB 60 00

0170 00 03 19 0E 09 01 0D 1A 14 06 0B 0F 12 1C 0C 18
0180 16 17 0A 05 10 07 1E 13 1D 15 11 2D 00 3A 00 00
0190 04 34 00 2C 00 24 31 25 2F 32 39 31 2C 23 3C 3D
01A0 36 37 33 21 2A 30 35 27 26 38 2E 2B 24 3E 24 39
```

Bild 3. Codeumwandlungstabelle für das Baudot-Unterprogramm

Wenn ein Rechnersystem während des Programmlaufs nichtnumerische Sachverhalte zu speichern hat, so ist es üblich, ausgesuchten Variablen die Funktion eines „Flags" zuzuweisen, die Aufgabe einer Merkzelle also, und in dieser Variable dann verschlüsselte Kurzzeichen einzulesen, die den zu speichernden Sachverhalt symbolisch abbilden.

Hans-Georg Joepgen

„Bit-Flags" ermöglichen elegante Basic-Wege

Bei Bedarf fragt das Programm dann diese Flags ab und macht Entscheidungen über die weitere Verfahrensweise von deren Inhalt abhängig. So kann der Programmierer beim Schreiben eines Spiels Mensch gegen Rechner beispielsweise eine Variable SF zum „Spielerflag" deklarieren und festsetzen, wenn der Inhalt von SF gleich eins sei, so ist der Rechner am Zuge, bei SF gleich Null jedoch der menschliche Gegner. Oder aber in einem kaufmännischen Zwecken dienenden Programm. Da erhält der Rechner in einem Flag „Kunden-Art" KA$ mitgeteilt, zu welcher Kundenart der Besteller einer Ware gehört und welche Rabattsätze bei der Preisgestaltung mithin anzuwenden oder welche Zahlungsbedingungen zu gewähren sind. Wenn KA$ = „G", dann geht's um den Auftrag eines Großhändlers, B für Behördenkunden, E für Einzelhändler; daß er eine Endverbraucherrechnung schreiben muß, erfährt der Rechner schließlich beim Abfragen des Kundenart-Flags, wenn er dort beispielsweise das Zeichen V findet. Denkbare Erweiterungen könnten in der gleichen Stringvariablen noch ein „S" für säumige Zahler enthalten sowie ein drittes Zeichen, etwa für Auslandskunden, das angibt, in welcher Sprache mit dem Geschäftsfreund zu korrespondieren ist. Man sieht: Sowohl numerische Variable als auch Zeichenketten-Variable bieten nahezu unbegrenzte Möglichkeiten zur Sachverhalts-Abbildung; entsprechend oft findet man die Verwendung solcher Flags in Programmen.

Weit weniger genutzt wird jedoch eine dritte Flagart in Basic-Programmen, wiewohl ihr Einsatz bei zahlreichen Aufgabenstellungen handfeste Vorteile bietet: Das „Bit-Flag". Es entspricht auf den ersten Blick einem numerischen Flag, mit dem Unterschied freilich, daß man hier zweckmäßigerweise nicht die Variablenart REAL benutzt, sondern die (durch ein dem Variablennamen folgendes Prozentzeichen gekennzeichnete) Variablenart INTEGER. Der PET und verwandte Maschinen stellen für diese Variablenart 16 bit zur Verfügung, von denen eines für das Vorzeichen der Zahl reserviert bleibt.

Hauptvorzug des Bit-Flag-Wortes: In einer einzigen Variablen können ohne Schwierigkeit 15 verschiedene Sachverhalte völlig unabhängig voneinander und in komprimierter Form gespeichert werden. Dies macht einerseits die Programmierung transparent und erleichtert zum zweiten die Fehlersuche im unfertigen und die Wartung des fertigen Programms. Bei der Informationsübergabe an das Flagwort lassen sich entweder Einzelbits, zusammenhängende Bitgruppen oder auch das Gesamtwort setzen – alles durch ein einziges Basic-Statement.

Zum Zugriff auf das Einzelbit könnte man sich eine Tabelle anfertigen, die den Stellenwert der Einzelbits in dezimaler Schreibweise enthält und die dann so aussehen würde:

Bit	0	1	2	3	4	5
Wert	1	2	4	8	16	32
Bit	6	7	8	9	10	
Wert	64	128	256	512	1024	
Bit	11	12	13	14	15	
Wert	2048	4096	8192	16384	32768	

Sollen im Flagwort beispielsweise nur die Bits 2 und 4 gesetzt, alle übrigen aber rückgesetzt werden, so würde man wie folgt programmieren:

```
F% = 4 + 16
```

Ein anderes Beispiel: Flagbit 10 soll gesetzt, der übrige Zustand des Wortes aber unverändert gelassen werden – dies geschieht so:

```
F% = F% OR 1024
```

Auch die AND-Funktion bietet elegante Möglichkeiten zur gruppenweise Bitmanipulation. So löscht

```
F% = F% AND 127
```

alle Bits mit den Ordnungsnummern 7 und größer.

Einzelbits werden wie folgt zum „Weichensteller" (Program Switch) gemacht, hier vorgeführt am Bit 2, das – wenn gesetzt – einen Sprung zur Subroutine 100 veranlaßt:

```
10 IF FG% AND 4 THEN GOSUB 100
```

Lästig ist hier noch ein wenig, daß man sich der Tabelle bedienen oder den Stellenwert der Einzelbits im Kopfe haben muß. Nun kann man aber leicht – und dies gibt dem „Bitflagging" erst die volle Praxisreife – die Ermittlung des Stellenwertes dem Rechner selbst übertragen und unmittelbar die Ordnungszahl des betreffenden Bits verwenden – und zwar dadurch, daß man anstatt der betreffenden Zahl in das jeweilige Statement den entsprechenden Potentausdruck einsetzt – nämlich zwei hoch Ordnungszahl (für das Exponentenzeichen, das von Maschine zu Maschine verschieden ist, z. B. Aufwärtspfeil beim PET: Dafür ist im untenstehenden Beispiel das Markzeichen M eingesetzt):

```
10 IF (FG% AND 2 M 7) OR (FG% AND
2 M 12) THEN...
```

stellt eine gezielte Abfrage der Bits 7 und 12 dar: Man muß sich bei dieser Programmiertechnik überhaupt nicht mehr darum kümmern, welchen Dezi-

malwert die einzelnen Stellen im Doppelbyte-Wort überhaupt repräsentieren. Ähnlich einfach gestaltet sich mit diesem Verfahren das Setzen einzelner Bits:

FG% = FG% OR 2 M 7
setzt Bit 7.

Bitflagging als Programmiertechnik in Basic empfiehlt sich noch durch einen weiteren Vorzug, der vor allem dann voll zum Tragen kommt, wenn den Sachverhalten, die durch die gesetzten oder gelöschten Einzelbits abgebildet werden, von Haus aus eine natürliche Priorität zukommt. Dann kann man durch eine arithmetische Abfrage zu einer weiteren Komprimierung des Programms kommen:

10 UF FG% > 2 M 9 THEN...

führt in jedem Fall zur Verzweigung, wenn von den Bits mit Ordnungsnummer 10 oder größer auch nur ein einziges gesetzt ist.

Man sieht, es lohnt sich auch für Hochsprachen-Programmierer, gelegentlich auf Bit-Ebene zu denken und somit in Gefilde abzusteigen, die sonst allein der Tummelplatz von Maschinensprache-Spezialisten und Assembler-Fans sind.

Freilich: „Bit-Flagging" funktioniert nicht auf allen Microcomputern. Erfolgreich waren Versuche auf dem PET 2001 und einer Maschine PC 100 mit Interpreter für Siemens-Basic und TRS-80, Fehlanzeige gab's auf Maschinen ITT 2020 PALSOFT und dem Apple-II-Plus.

Literatur

[1] *Wirth, Niklas:* „Algorithmus plus Data Structures = Programs", Prentice-Hall Incorporated, Englewood Cliffs, New Jersey, USA, 1976

[2] *Weems, Chip:* „Designing Structured Programs" in „Byte", Heft 8, Band 1978, Byte Publications – Mc Graw Hill, Peterborrough, New Hampshire, USA

[3] *Joepgen, Hans-Georg:* „PET als listenreicher Kartenspieler" in „Hobbycomputer", Sonderheft der Zeitschriften Elektronik, Funkschau und Elo, Franzis-Verlag, München, 1978 und 1979.

[4] *Joepgen, Hans-Georg:* „Wanzenjagd – über die Fehlersuche in Basic-Programmen" in „Hobby-Computer 2", Sonderheft der Funkschau, Franzis-Verlag, München 1979. ■

TRS-80 macht Musik

Ohne zusätzliche Hardware kann man mit dem TRS-80 Musik machen. Dazu ist das in *Bild 1* gezeigte Basic-Programm zu laden und zu starten (MEMORY SIZE = 20400). Anschließend lädt man das Programm von *Bild 2*. Läßt man es ohne Änderung laufen, wird die Melodie von „Hänschen klein" ausgegeben.

Als Ausgabeport dient der Kassettenausgang. Das Signal kann entweder einem Verstärker zugeführt oder direkt vom Kassettenrecorder aufgenommen werden.

Will man eine eigene Melodie komponieren, muß man die Zeilen 74 und 76 ändern. In Zeile 74 ist der Variablen D$ ein String zugewiesen, der nur aus Ziffern besteht. Jede einzelne Ziffer entspricht der Notendauer (1 = kurz, 9 = lang). Die Noten selbst stehen in der DATA-Zeile 76, die mit END abgeschlossen wird.

Insgesamt können 26 Töne erzeugt werden, die je einen Halbton auseinander liegen. Das entspricht gut zwei Oktaven. Der tiefste Ton wird mit C bezeichnet, es folgen CI (≙ cis), D, DI (dis) usw. (*Bild 3*). Die nächsthöhere Oktave wird mit dem Zusatz 1 gekennzeichnet, also C1, CI1, D1 usw. Da mit dem letzten Ton wieder eine neue Oktave beginnt, erhält er den Zusatz 3. Ändert man die DATA-Zeile entsprechend ab, kann man auch auf andere Tonarten übergehen (Beispiel: D, DI, E, EI, F, G usw.).

Das Programm läuft mit Level-II-Basic (4 KByte). Es basiert auf einem Maschinenprogramm [1], das mit Hilfe von POKE-Befehlen (Bild 1) in den Speicher gebracht und mit USR(0) aufgerufen wird. Ist das Hauptprogramm (Bild 2) einmal gelaufen, kann die Tonfolge mit USR(0) beliebig oft abgerufen werden.

Rudolf Hofer

Literatur

[1] *Pilgrim, P.:* System/Command. 80-US, Mai/Juni 79, S. 30...31.


```
5 POKE 16526,2o5: POKE 16527,79
1o CLEAR1ooo:DIM TN(26),T$(26):S$="":V=VARPTR(S$)
2o V1=INT(V/256):Vo=V-V1x256:REM V IN BYTE-GROESSE
3o POKE 2o478,Vo:POKE 2o479,V1:REM V NACH SBUFF
4o FOR T=1 TO 26: REM HARMONISCHE TONLEITER
5o F=o.oo4x1.o5946 T:P=INT(1/F):TN(T)=P:NEXT
52 FOR I=1 TO 26: READ T$(I):NEXT I
54 DATA C,CI,D,DI,E,F,FI,G,GI,A,AI,H,C1,CI1,D1,DI1,E1,F1,FI1,G1,GI1,A1,A
   I1,H1,C2,CI2
6o P$=""
62 FOR J=1 TO 255
64 READ A$: IF A$="END" THEN 74
66 FOR I=1 TO 26
68 IF A$=T$(I) THEN GOTO 71
7o NEXT I
71 P$=P$+CHR$(64+I)
72 NEXT J
74 D$="8448444444448"
76 DATA G,E,E,F,D,D,C,D,E,F,G,G,G,END
8o FOR I=1 TO LEN(D$): REM TONFOLGE
9o DU=(ASC(MID$(D$,I,1))-48)/2: IF DU=o THEN DU=1
1oo S$=S$+CHR$(DUx2o)+CHR$(TN(ASC(MID$(P$,I,1))-64))
11o NEXT: K=USR(o): REM SPIELE TONFOLGE IN S$
```

Bild 2. Durch Ändern der Zeilen 74 und 76 können Melodien selbst komponiert werden

```
1 FOR B=2o42o TO 2o477:READ A:POKE B,A:NEXT
2 DATA 33,2o5,79,34,142,64,195,25,26,42,254,79,126,183,245,35,94,35,86,235,241,2oo,61,2oo,61,245,86,3o,o,35
3 DATA 78,35,62,1,2o5,244,79,4o,237,62,3,2o5,244,79,32,242,24,228,211,255,65,27,122,179,2oo,16,25o,2o1
```

Bild 1. Nach dem Einschalten wird „MEMORY SIZE?" mit 20400 beantwortet, dann lädt man dieses Programm und startet es (RUN) – es handelt sich um das Maschinenprogramm zur Tonerzeugung [1]

Der auf dem Mikrocomputer AIM-65 (bzw. PC-100) vorhandene Thermodrucker wird rein per Software angesteuert. Auf dem Druckkopf befinden sich nebeneinander zehn kleine Heizelemente, die bei geeigneter Ansteuerung und der durch die Mechanik vorgegebenen Zickzack-Bewegung des Kopfes beliebige Symbole ausgeben können, so auch Grafiken mit einer horizontalen Auflösung von 100 Punkten.

Thomas Müller, Andreas Koch

Plotten mit dem AIM-Drucker

Die hundert adressierbaren Punkte einer Druckzeile können dezimal eingegeben werden; das Programm in *Bild 1* enthält daher unter anderem eine entsprechende Umrechungsroutine und steuert den Drucker so an, daß maximal hundert Kurven gleichzeitig geplottet werden könnten (!).

Das Programm holt sich für jede auszugebende Zeile die dezimalen Werte aus dem ab der Adresse 0305 stehenden Anwenderprogramm. Die Punkte sind von 00 (links außen) bis 99 (rechts außen) ab der Zero-Page-Speicherzelle 0051 in beliebiger Reihenfolge adressierbar. Das Ende jeder Zeile wird mit 0A gekennzeichnet. Leerstellen benötigen keine Speicherplätze. Wenn der Plotvorgang beendet werden soll, braucht man nur vom Anwenderprogramm aus in die Speicherzelle 0000 den Wert Null zu schreiben; diese Zelle kann daher z. B. als Zeilenzähler verwendet werden.

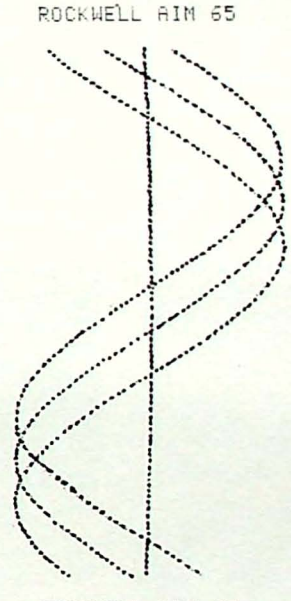
Das Programm synchronisiert den Drucker bei jedem Zeilenbeginn neu auf das Prozessor-Timing. Sollten sich Probleme in bezug auf eine „wirre" Lage der ausgegebenen Punkte ergeben, so läßt sich dies meist mit dem Trimmer VR3 auf der AIM-Platine beheben. Außerdem läßt sich die Pause zwischen zwei Punkten in 024D, die Pause zwischen einer Neusynchronisation und dem Plotten des ersten Punktes in 028A und die Ansteuerzeit der Thermoelemente in 0267 individuell ändern. (Vorsicht: Bei Programmierfehlern kann der Drucker durch eine Überhit-

```
0200 A9 0A 85 0C | 02A0 20 A9 00 95
0204 AD 0D A8 29 | 02A4 00 CA D0 FB
0208 10 F0 F9 20 | 02A8 A2 01 A9 40
020C 87 02 AD 00 | 02AC 85 1B A9 00
0210 A8 60 A2 00 | 02B0 85 1A B5 50
0214 A9 00 85 0A | 02B4 C9 0A F0 3E
0218 85 0B 18 8A | 02B8 29 F0 EA EA
021C F8 69 01 AA | 02BC 18 4A 4A 4A
0220 D8 18 36 10 | 02C0 4A 85 1C F8
0224 36 00 90 15 | 02C4 69 01 8D E9
0228 B5 30 8D 33 | 02C8 02 8D EB 02
022C 02 8D 35 02 | 02CC 69 10 8D EF
0230 B5 40 05 0B | 02D0 02 8D F1 02
0234 85 0B E0 10 | 02D4 D8 B5 50 29
0238 D0 E0 4C 4A | 02D8 0F F0 0B 85
023C 02 EA EA EA | 02DC 1D 18 26 1B
0240 EA EA EA EA | 02E0 26 1A C6 1D
0244 EA EA EA 4C | 02E4 D0 F7 A5 1A
0248 36 02 A2 06 | 02E8 05 06 85 06
024C A0 44 88 D0 | 02EC A5 1B 05 16
0250 FD CA D0 F8 | 02F0 85 16 E8 4C
0254 AD 00 A8 05 | 02F4 AA 02 20 00
0258 0A 8D 00 A8 | 02F8 02 A5 00 D0
025C AD 01 A8 05 | 02FC 9F A9 E1 8D
0260 0B 8D 01 A8 | 0300 0C A8 4C 82
0264 A2 02 A0 A0 | 0304 E1 A6 57 BD
0268 88 D0 FD CA | 0308 00 04 85 51
026C D0 F8 A9 FF | 030C E8 E0 78 D0
0270 45 0A 2D 00 | 0310 02 A2 00 86
0274 A8 8D 00 A8 | 0314 57 A6 56 BD
0278 A9 FF 45 0B | 0318 00 04 85 52
027C 2D 01 A8 8D | 031C E8 E0 78 D0
0280 01 A8 C6 0C | 0320 02 A2 00 86
0284 D0 8C 60 A2 | 0324 56 A6 58 BD
0288 0A A0 30 88 | 0328 00 04 85 53
028C D0 FD CA D0 | 032C E8 E0 78 D0
0290 F8 4C 12 02 | 0330 02 A2 00 86
0294 20 00 02 A9 | 0334 58 A9 50 85
0298 C1 8D 0C A8 | 0338 54 A9 0A 85
029C 20 05 03 A2 | 033C 55 C6 00 60
```

Bild 1. Hex-Dump des Plotter-Programms (0200...0304) und eines Anwenderprogramms (0305...033F), das zusammen mit den Daten in Bild 2 drei Sinuskurven ausdruckt; Startadresse ist 0297

```
0030 00 0A 0A 0B
0034 0B 0B 0B 0B
0038 0B 0B 00 00
003C 00 00 00 00
0040 0B 02 01 80
0044 40 20 10 08
0048 04 02 00 00
004C 00 00 00 00
0050 01 70 89 44
0054 50 0A 00 0A
0058 14 1A 0B 18
0400 50 52 55 57
0404 60 62 65 67
0408 70 72 74 76
040C 78 81 82 84
0410 86 88 89 91
0414 92 93 95 96
0418 97 97 98 98
041C 99 99 99 99
0420 99 98 98 97
0424 97 96 95 93
0428 92 91 89 88
042C 86 84 82 81
0430 78 76 74 72
0434 70 67 65 62
0438 60 57 55 52
043C 50 47 44 42
0440 39 37 34 32
0444 29 27 25 23
0448 21 18 17 15
044C 13 11 10 08
0450 07 06 04 03
0454 02 02 01 01
0458 00 00 00 00
045C 00 01 01 02
0460 02 03 04 06
0464 07 08 10 11
0468 13 15 17 18
046C 21 23 25 27
0470 29 32 34 37
0474 39 42 44 47
10C  4C 97 02 00
```

◄ Bild 2. Datentabellen für das Programm von Bild 1. Ab der Adresse 0400 stehen die Amplitudenwerte der Sinuskurven; der F1-Vektor bei 010C ist für die Startadresse 0297 programmiert



◄ Bild 3. So sieht die Grafik aus, die der AIM-65 unter Zuhilfenahme von Plotterprogramm und Sinus-Wertetabelle ausgibt

```
<D>
FROM=30 TO=58
OUT=T F=PLOTD T=1
MORE?Y
FROM=200 TO=33F
MORE?Y
FROM=400 TO=477
MORE?Y
FROM=10C TO=10E
MORE?N
```

Bild 4. Protokoll der Aufzeichnung aller Programmteile auf Kassette

zung der Thermoelemente beschädigt werden; wenn er „schwarzes Papier" ausstößt, sofort Reset drücken!)

Das alles klingt zunächst ein wenig kompliziert, wird aber gleich deutlich, wenn wir noch die Datentabellen von *Bild 2* eingeben. Zusammen mit dem Plotprogramm (0200...0304) und dem in Bild 1 ebenfalls schon aufgelisteten einfachen Anwenderprogramm (0305...033F) druckt uns der AIM-65 beim Start an der Adresse bzw. beim Betätigen der Taste F1 drei gegeneinander phasenverschobene Sinuskurven (*Bild 3*). Wie schon erwähnt, können leichte Punktverschiebungen mit VR3 ausgeglichen werden. *Bild 4* zeigt schließlich noch ein Protokoll der Abspeicherung aller aufgeführten Programmteile auf Kassette – hier mit dem File-Namen PLOTD wie „Plotter-Demonstration". ■

```
10 REM R.F.M. 1.12.79 QRB MITTELS QTH-KENNER
20 REM
30 CLS:P."EIGENER";:GOS.500
80 A(20)=L:A(21)=B
100 P."FREMDER";:GOS.500
110 M=A(20):C=A(21)
120 M=M-L:M=M*M:C=C-B:C=C*C
150 L.X=A(21):GOS.30360:L.L=Y:L.X=B:GOS.30360:L.X=C+M*L*Y:GOS.30030
200 P.:P."ENTFERNUNG=";INT(111.2*Y+.5);" KM":P.:G.100
500 A=1:B=2:C=3:D=4:E=5:F=6:G=7:H=8:I=9:J=10:K=11:L=12:M=13
510 N=14:O=15:P=16:Q=17:R=18:S=19:T=20:U=21:V=22:W=23:X=24:Y=25:Z=26
520 IN." QTH-KENNER (B,B,ZZ,B) ";A(1),A(2),A(3),A(4)
530 IF (A(1)<A)+(A(1)>Z)+(A(2)<A)+(A(2)>Z)+(A(4)<A)+(A(4)>J) T.800
540 IF A(1)>U T.A(1)=A(1)-Z
550 A(1)=A(1)-1:A(2)=A(2)-1
560 IF (A(3)<1)+(A(3)>80)T.800
570 REST.
580 DATA 2,3,3,3,3,2,3,1,2,1,1,1,1,2,1,3,2,2,2,2
590 F.I=1T.A(4):READ B:READ C:N.I
610 A=INT(A(3)/10):D=A(3)-10*A:IF D=0 T.D=10:A=A-1
620 D=D-1
700 L=A(1)+A(1)+(B+D+D+D)/15-0.0333333
710 B=(C-A-A-A)/24+A(2)+40.854167
730 P."LAENGE= ";L;" GRAD  / BREITE= ";B;" GRAD"
750 RET.
800 P."PFUI  !":G.520
30030 IFX=0T.Y=0:RET.
30060 Y=X/2:Z=0
30070 W=(X/Y-Y)/2:IF(W=0)+(W=Z)T.RET.
30090 Y=Y+W:Z=W:G.30070
30360 W=ABS(X)/X:X=X+90:GOS.30376:IF(Z=-1)*(W=1)T.Y=-Y
30365 RET.
30376 Z=ABS(X)/X:X=Z*X
30380 IF X>360T.X=X-360
30390 IF X>90T.X=X/90:Y=INT(X):X=(X-Y)*90:ONYG.30410,30420,30430
30400 X=X/57.29578:IF ABS(X)<2.48616E-4 T.Y=0:RET.
30405 G.30440
30410 X=90-X:G.30400
30420 X=-X:G.30400
30430 X=X-90:G.30400
30440 Y=X-X*X*X/6+X*X*X*X*X/120-X*X*X*X*X*X*X/5040
30450 Y=Y+X*X*X*X*X*X*X*X*X/362880:IF Z=-1T.Y=-Y
30455 RET.
```

Entfernungsberechnung mit TRS-80-Level 1

In mehreren Publikationen – so auch im Sonderheft „Hobbycomputer 2" der FUNKSCHAU – sind Entfernungsberechnungsprogramme auf der Grundlage der im Amateurfunkdienst gebräuchlichen QTH-Kenner veröffentlicht worden. Für einen der meistverkauften Hobbycomputer, dem TRS-80 mit Level-1-Basic, wurde ein solches Programm jedoch noch nicht veröffentlicht. Die Gründe sind: Bei Level-1-Basic ist eine Stringbehandlung auf Einzelzeichenbasis nicht vorgesehen; Level-1-Basic enthält keine trigonometrischen Funktionen und keine Funktion zum Wurzelziehen; und Level-1-Basic-Programme können mit Standardsoftware nicht ausgedruckt werden.

Diese Probleme wurden wie folgt gelöst: Die Umcodierung der alphanumerischen Eingabe erfolgt durch Nutzung der Tastaturvariablen. Die Tastaturvariablen werden in den Zeilen 500 und 510 zugewiesen. Die Eingabe des QTH-Kenners FN31a erfolgt als F,N,31,A (Enter). Das Programm behandelt den QTH-Kenner als Variable A(1), A(2), A(3) und A(4). Die benötigten Funktionen für SQR und COS stehen als Basic-Unterprogramme ab Zeile 30030. Es handelt sich hier um von Tandy angegebene Unterprogramme, die geringfügig modifiziert sind. Zum Ausdruck des Programmes wurde ein Assemblerprogramm verwendet. Als Drucker diente der H14 von Heathkit.

Ing.(grad.) R.-F. Matthaei

Stichwortsuche in BASIC

In FUNKSCHAU 1978, Heft 24, war unter dem Titel „KIM auf Datensuche" ein Stichwort-Suchprogramm beschrieben, das sich ideal für eine Software-Kartei für Adressen, Telefonnummern oder Literaturverzeichnisse eignet. Selbstverständlich läßt sich ein solches Programm auch in BASIC erstellen, wenn man die deutliche langsamere Suchgeschwindigkeit in Kauf nimmt. Die Daten, in denen nach dem einzugebenden Stichwort gesucht werden soll, sind als Strings in DATA-Blöcken (im *Bild* bei der Zeilennummer 500) zusammengefaßt. Wird ein Stichwort irgendwo innerhalb eines Strings gefunden, so wird der gesamte String ausgegeben. Bei einem Telefonnummern-Verzeichnis könnte der Data-Block etwa so aussehen:

500 DATA HUBER 1234567, MUELLER 7654321, MEIER 432123 usw.

Das Programm zeigt recht deutlich die Anwendung von MID $ (D $, I, K). I gibt dabei den Ort innerhalb des Strings an, wo gesucht wird, und K die Länge des Buchstaben-Vergleichs, die mit der Stichwort-Länge identisch ist. Fe.

Dieses BASIC-Programm – es wurde auf einem AIM-65 entwickelt – sucht in DATA-String-Blöcken nach einem einzugebenden Stichwort und eignet sich daher recht gut als Karteiprogramm

```
80 RESTORE:INPUT"STICHWORT";A$:K=LEN(A$)
90 READD$:IF D$="END" THEN 80
100 S=LEN(D$)-K:IF S<0 THEN 90
110 FOR I=1 TO S+1:M$=MID$(D$,I,K)
120 IF M$=A$ THEN 150
130 NEXT
140 GOTO90
150 PRINTD$:GOTO 90
500 DATA**MIT END ABSCHLIESSEN!**
READY.
```

Das hier wiedergegebene Programm für 6502-Mikrocomputer stellt einen sehr komfortablen, zeigerorientierten Editor dar, mit dem sich Briefe, Tabellen und beliebige andere Texte in den Speicher schreiben, redigieren und wieder auslesen lassen. Die Adressenbelegung ist für den KIM-1 ausgelegt. Eine Adaption an andere Systeme ist sehr einfach, da nur zwei Befehle geändert werden müssen.

Dipl.-Phys. H. M. Ihme

6502-Texteditor

Ein Editor ist ein Programm zum Erstellen, Verändern und Durchsuchen von Texten, Dateien und Programmen. Der Mikrocomputer KIM muß mit einer Speichererweiterung von mindestens 4 KByte und einem Datensichtgerät ausgestattet sein. Benötigt werden neben den 2 KByte für das Programm selbst (2000-27FF) ein Bereich für den Eingangsspeicher (200-3FF), ein Bereich für den Hauptspeicher, der möglichst groß sein sollte (2800-2EFF), und ein Bereich für einen Hilfsspeicher (2F00-2FFF). Außerdem werden die Adressen B0...D2 der Zero Page vom Editor benutzt, ebenfalls ein Teil des Stacks auf Seite 1.

Der Editor ist zeigerorientiert, d. h. fast alle Befehle werden relativ zur Stellung eines Zeigers ausgeführt oder dienen zum definierten Verschieben eines Zeigers. Eine Reihe von Befehlen erfordert die Angabe einer Zahl, die im folgenden n genannt wird. Diese Zahl wird dezimal behandelt und kann zwischen −9999 und + 9999 liegen. Wird n nicht angegeben, so wird n=0 gesetzt. Ist n größer oder kleiner als die angegebenen Werte, so werden nur die letzten 4 Stellen berücksichtigt.

Im folgenden soll der leistungsfähige Befehlssatz des Editors im einzelnen besprochen und anhand von Beispielen erläutert werden. Anschließend erfolgt noch eine Kurzübersicht der Befehle. Es können alle ASCII-Zeichen editiert werden, bis auf die 5 Zeichen ESCAPE (1B), CNTRL B (02), CNTRL R (12), CNTRL X (18) und DEL (7F), die eine Sonderstellung im Befehlssatz einnehmen. Das Zeichen ESCAPE dient dem Editor zum Unterscheiden von Zeichen- und Befehlsketten und, wenn es doppelt vorkommt, als Befehl, die im Eingangsspeicher abgelegte, gemischte Befehls- und Zeichenkette abzuarbeiten (DELIMITER). ESCAPE wird als $ „geechot", darf aber mit $ = hex 24 nicht verwechselt werden.

CNTRL X (CANCEL) löscht die gesamte Eingabe. DEL löscht das zuletzt eingegebene Zeichen und „echot" es. CNTRL R zeigt die letzte eingegebene Zeile an. ESCAPE, CNTRL X, CNTRL R und DEL beziehen sich also nur auf den Eingabespeicher, nicht auf Haupt- und Hilfsspeicher. Es sei noch darauf hingewiesen, daß die Taste „Return" als Zeilentrenner behandelt wird. Line Feed wird bei der Ausgabe automatisch eingefügt.

Der KIM-Editor hat 2 Startadressen: Ein Start bei 2000 löscht Haupt- und Hilfsspeicher (Kaltstart); ein Start bei 2003 setzt voraus, daß ein Text im Hauptspeicher (und gegebenenfalls auch im Hilfsspeicher) vorhanden ist (Warmstart). Nach dem Start meldet sich der Editor mit einem „!" (Prompt) zum Zeichen dafür, daß mit der Eingabe begonnen werden kann. Es kann jetzt eine beliebige (sinnvolle) Mischung aus Befehls- und Zeichenketten eingetippt werden, die nach der Eingabe von 2 ESCAPES abgearbeitet wird.

Die Befehle

nT (TYPE)
gibt n Zeilen vom Zeiger an aus. Wird n als 0 eingegeben oder ganz weggelassen, so wird die laufende Zeile ausgegeben, also die Zeile, in der der Zeiger steht. Außerdem wird vor dem Zeichen, auf das der Zeiger zeigt, ein (^) bei der Ausgabe eingefügt. Wird n als # eingegeben, dann wird der gesamte Hauptspeicherinhalt gezeigt, ohne daß die Stellung des Zeigers verändert wird.

```
Beispiel: !T$$
          (^)
          !
```

Da kein Text im Hauptspeicher vorliegt, steht der Zeiger am Anfang des Speichers.

ITEXT$ (Insert)
Vor dem Zeiger wird der Text TEXT im Hauptspeicher eingefügt. Alle Zeichen nach dem Zeiger werden entsprechend verschoben. Die relative Stellung des Zeigers bleibt unverändert. Der Befehl muß mit einem ESCAPE abgeschlossen werden, damit der Editor erkennt, daß hier der Text zu Ende ist und die nächsten Zeichen wieder als Befehle zu interpretieren sind.

```
Beispiel: !IZEILE    (Carriage-Return)
          $T$$
          (^)
          !
```

Der Zeiger steht hinter „Return" am Anfang der nächsten Zeile.

```
          !-1T$$
          ZEILE
          !
```

Der Befehl -1T zeigt die vorige Zeile.

Eine Reihe von Befehlen gestattet es, den Zeiger zeichen- oder zeilenweise hin- und herzuschieben oder an den Anfang oder Ende des gesamten Textes zu setzen.

J
Zeiger an den Anfang des Textes setzen.

```
Beispiel: !JT $$
          (^)ZEILE
          !
```

Z
Zeiger an das Ende des Textes setzen.

```
Beispiel: !ZI2.ZEILE
          $-2T$$
          ZEILE
          2. ZEILE
          !
```

Der Zeiger steht immer noch am Ende des Textes, es wurde der neue Text 2.ZEILE eingefügt.

nM (MOVE)
bewegt den Zeiger um n Zeichen vorwärts oder rückwärts, je nachdem, ob n positiv oder negativ ist.

```
Beispiel: !-3MT$$
          2. ZEI(^)LE
          !
```

(auch der Carriage-Return ist ein Zeichen!)
0M bewegt den Zeiger nicht, ist also ein leerer Befehl.
nL (LINE)
bewegt den Zeiger um n Zeilen vor oder zurück. Wenn n=0 ist, wird der Zeiger an den Anfang der laufenden Zeile gesetzt.

Beispiel: !OLT$$
(^)2. ZEILE
!

Zwei Befehle ermöglichen es, Zeichen oder Zeilen zu löschen:

nD (DELETE)
löscht n Zeichen vom Zeiger an vorwärts oder rückwärts. 0D ist wie 0M ein leerer Befehl.

Beispiel: !2D#T$$
ZEILE
ZEILE
!

Die Zeichen 2. wurden gelöscht, der Zeiger steht in der 2. Zeile vor ZEILE.

nK (KILL)

löscht n Zeilen vom Zeiger an in beiden Richtungen. Man beachte: 0K löscht die laufende Zeile vom Anfang bis zum Zeiger, 1K löscht die laufende Zeile vom Zeiger an bis einschließlich des nächsten Carriage-Returns.

Beispiel: !3M0KT$$
(^)LE
!

Wird n= # gesetzt, dann wird der gesamte Hauptspeicher gelöscht. Im vorliegenden Beispiel wird also #K die restlichen Zeichen löschen und den Zeiger an den Anfang des Hauptspeichers stellen.

nX (n mal)
gestattet es, die nachfolgende Kette aus Befehls- und Textzeichen n-mal auszuführen. Die Zahl n wird hier immer positiv behandelt, auch wenn sie negativ ist.

Beispiel: !5XI.ZEILE$$
!§T$$
.ZEILE
.ZEILE
.ZEILE
.ZEILE
.ZEILE
!

Die Verwendung dieses Befehls ist einigen Einschränkungen unterworfen. Der Befehl darf in einer Befehlskette nur einmal vorkommen und bezieht sich dann immer auf alle Befehle, die hinter dem X stehen. Wird X mehrmals in einer Kette verwendet, dann werden nur die Befehle nach dem letzten X n-mal ausgeführt, die anderen jeweils nur einmal. Wird 0X verwendet, dann werden die Befehle danach überhaupt nicht behandelt. Außerdem darf der X-Befehl nur maximal mit 250 Zeichen (dezimal) nach dem PROMPT (!) benutzt werden.

STEXT$ (SEARCH)
sucht vom Zeiger an den gesamten Hauptspeicher nach dem Wort TEXT durch. TEXT muß wie beim I-Befehl mit einem ESCAPE abgeschlossen werden. Wird TEXT gefunden, dann steht der Zeiger hinter dem letzten gefundenen Zeichen (hier das 2.T von TEXT). Wird TEXT nicht gefunden, dann erfolgt die Fehlermeldung

Nicht gefunden!

und der Zeiger wird an den Anfang des Hauptspeichers gestellt. Die Abarbeitung der Befehlskette wird abgebrochen.

Beispiel: !JI1$1LI2$1LI3$1LI4$
1LI5$JS3$T$$
3(^).ZEILE
!

Die Zeilen wurden mit Zeilennummern versehen, die 3 von der 3.ZEILE wurde gesucht.

TEXT darf max. 255 Zeichen lang sein und auch Carriage-Returns enthalten. Man kann nur vorwärts suchen.

CTEXT1$TEXT2$ (Change)
sucht vom Zeiger an den gesamten Hauptspeicher durch nach dem Wort TEXT1 und ersetzt es durch das Wort TEXT2.

```
2000 4C 29 23 4C 32 23 02 00 03 FF 28 00 2E FF 2F 00
2010 2F FF A5 C9 85 B0 A5 CA 85 B1 60 E6 B4 D0 02 E6
2020 B5 60 A5 B6 85 BE A5 B7 85 BF 60 A5 BE 85 B6 A5
2030 BF 85 B7 60 20 AF 21 20 08 22 B0 0E 24 C0 10 0B
2040 20 EE 21 B0 05 20 7F 21 90 F6 60 20 FA 21 B0 FA
2050 20 6D 21 90 F6 60 A0 07 B9 64 20 20 25 27 88 10
2060 F7 4C 47 23 2A 2A 65 72 6F 43 2A 2A 20 D1 21 A5
2070 B4 85 C9 AD B5 00 85 CA A0 FF 84 C8 38 A5 B2 E5
2080 B0 85 B8 A5 B3 E5 B1 85 B9 18 A5 B8 65 B4 85 B6
2090 48 A5 B9 65 B5 85 B7 48 38 AD 0D 20 E5 B6 AD 0C
20A0 20 E5 B7 90 B1 E6 C8 E6 B8 E6 B9 38 A5 B4 E5 B0
20B0 A5 B5 E5 B1 A2 00 90 02 A2 02 A1 B0 81 B4 90 14
20C0 C6 B2 98 45 B2 D0 02 C6 B3 C6 B6 98 45 B6 D0 02
20D0 C6 B7 B0 06 20 24 21 20 1B 20 C6 B8 D0 02 C6 B9
20E0 D0 D2 20 DA 21 A5 C8 10 05 68 68 38 B0 07 68 85
20F0 B3 68 85 B2 18 A2 00 60 20 57 21 B0 0E A1 B6 48
2100 20 41 21 68 C9 23 D0 02 18 60 38 60 A2 03 A9 00
2110 95 C2 CA 10 FB E8 85 C8 60 38 A5 B0 E5 B2 A5 B1
2120 E5 B3 B0 06 E6 B0 D0 02 E6 B1 60 38 AD 0B 20 E5
2130 B0 AD 0A 20 E5 B1 B0 08 A5 B0 D0 02 C6 B1 C6 B0
2140 60 38 A5 B6 ED 09 20 E9 02 A5 B7 ED 08 20 B0 06
2150 E6 B6 D0 02 E6 B7 60 38 AD 07 20 E5 B6 AD 06 20
2160 E5 B7 B0 08 A5 B6 D0 02 C6 B7 C6 B6 60 38 A5 B8
2170 E5 B2 A5 B9 E5 B3 B0 06 E6 B8 D0 02 E6 B9 60 38
2180 AD 0B 20 E5 B8 AD 0A 20 E5 B9 B0 08 A5 B8 D0 02
2190 C6 B9 C6 B8 60 20 6D 21 B0 07 A1 B8 C9 0D D0 F5
21A0 18 60 20 7F 21 B0 07 A1 B8 C9 0D D0 F5 18 60 A5
21B0 B0 9D B8 00 A5 B1 85 B9 60 A5 B8 85 B0 A5 B9 85
21C0 B1 60 A2 01 B5 B8 B4 B0 95 B0 94 B8 CA 10 F5 E8
21D0 60 A5 B0 85 BA A5 B1 85 BB 60 A5 BA 85 B0 A5 BB
21E0 85 B1 60 A5 B0 45 B8 D0 04 A5 B1 45 B9 60 20 08
21F0 22 B0 06 E6 C1 D0 02 E6 C0 60 20 08 22 B0 FA A5
2200 C1 D0 02 C6 C0 C6 C1 60 A5 C0 05 C1 F0 02 18 60
2210 38 60 20 AF 21 20 08 22 90 07 20 A2 21 B0 EF 90
2220 EF 24 C0 30 0C 20 FA 21 B0 06 20 95 21 4C 25 22
2230 60 20 A2 21 B0 D8 20 EE 21 90 F6 60 A1 B6 38 E9
2240 30 90 05 C9 0A 90 02 18 60 48 B5 C3 95 C2 E8 E0
2250 04 90 F7 68 85 C5 A2 00 20 41 21 38 60 B9 01 00
2260 85 C0 B9 00 00 85 C1 F8 A0 10 8A A2 02 95 C2 CA
2270 10 FB 06 C1 26 C0 90 0B A2 02 B5 C2 69 00 95 C2
2280 CA 10 F7 88 F0 0F 18 A2 02 B5 C2 75 C2 95 C2 CA
2290 10 F7 4C 72 22 D8 84 C8 B9 C2 00 D0 04 46 C8 90
22A0 05 20 60 27 E6 C8 C8 C0 03 90 ED 20 4E 27 4C 5E
22B0 25 18 A5 C1 75 C2 85 C1 A5 C0 69 00 85 C0 60 A0
22C0 03 8A 85 C0 85 C1 85 C6 20 B1 22 A5 C1 85 C7 A5
22D0 C0 85 C6 06 C1 26 C0 06 C1 26 C0 18 A5 C1 65 C7
22E0 85 C1 A5 C0 65 C6 85 C0 06 C1 26 C0 E8 88 D0 D8
22F0 20 B1 22 A2 00 A5 C8 D0 01 60 38 8A E5 C1 85 C1
2300 8A E5 C0 85 C0 60 AD 0B 20 85 B2 AD 0A 20 85 B3
2310 A2 00 60 AD 0B 20 85 B0 AD 0A 20 85 B1 60 AD 07
2320 20 85 B6 AD 06 20 85 B7 60 20 3D 27 20 B6 27 4C
2330 47 23 20 06 23 20 BE 27 A1 B2 C9 FF F0 09 E6 B2
2340 D0 02 E6 B3 4C 38 23 D8 CA 9A E8 20 13 23 A9 1B
2350 85 D2 20 4E 27 A9 21 20 1C 27 20 1E 23 20 45 27
2360 C9 18 F0 EE C9 12 D0 42 A9 5E 20 25 27 A9 52 20
2370 25 27 20 22 20 A1 B6 C9 0D F0 08 20 57 21 90 F5
2380 20 4E 27 A5 B6 C5 BE A5 B7 E5 BF B0 D0 A1 B6 C9
2390 02 D0 06 20 F2 27 4C A2 23 C5 D2 D0 02 A9 24 20
23A0 1C 27 20 41 21 90 DC 4C 5D 23 C9 7F D0 1D 20 57
23B0 21 B0 9F A1 B6 C5 D2 D0 08 A9 24 20 1C 27 4C 5D
23C0 23 C9 02 D0 F6 20 F2 27 4C 5D 23 81 B6 C9 02 D0
23D0 03 20 F2 27 C9 0D D0 03 20 53 27 C5 D2 F0 08 20
23E0 41 21 B0 21 4C 5D 23 A9 24 20 25 27 20 57 21 90
23F0 03 4C 52 23 A1 B6 C5 D2 F0 16 20 41 21 20 41 21
2400 B0 03 4C 5D 23 A5 D2 81 B6 20 41 21 A5 D2 81 B6
```

```
241o 2o 4E 27 86 BC 86 BD 2o 1E 23 2o oC 21 A1 B6 C9
242o 2D Do o5 E6 C8 2o 41 21 2o 3C 22 Bo FB 2o BF 22
243o A1 B6 29 DF C9 58 Do 16 A5 C8 Fo o3 2o FA 22 A5
244o Co 85 BD A5 C1 85 BC A5 B6 85 CB 4C 49 25 C9 4A
245o Do o6 2o 13 23 4C 5E 25 C9 5A Do oB A5 B2 85 Bo
246o A5 B3 85 B1 4C 5E 25 C9 48 Do o3 4C oo 1C C9 54
247o Do o3 4C 67 25 C9 4D Do o6 2o 34 2o 4C 94 24 C9
248o 44 Do o3 4C FF 26 C9 49 Do o3 4C E5 25 C9 4C Do
249o o9 2o 82 27 2o B9 21 4C 5E 25 C9 4B Do o3 4C E3
24Ao 26 C9 53 Do o6 2o 59 26 4C 1A 24 C9 43 Do o3 4C
24Bo Bo 26 C9 56 Do 21 2o D1 21 8A 85 B4 85 B5 2o 13
24Co 23 A1 Bo C9 oD Do o3 2o 1B 2o 2o 19 21 9o F2 2o
24Do DA 21 Ao B4 4C 5D 22 C9 4E Do 12 38 A5 B2 ED oB
24Eo 2o 85 B4 A5 B3 ED oA 2o 85 B5 4C D2 24 C9 45 Do
24Fo 1o A5 B3 2o 6o 27 A5 B2 2o 6o 27 2o 4E 27 4C 5E
25oo 25 C9 42 Do 37 A5 Co 85 CE A5 C1 85 CF 2o 41 21
251o A1 B6 29 DF C9 43 Do o6 2o CD 27 4C 5E 25 C9 54
252o Do o9 2o CD 27 2o 76 27 4C E8 26 C9 4B Do o6 2o
253o B6 27 4C 5E 25 C9 52 Do F9 4C o8 27 C5 D2 Do 1E
254o 2o 41 21 A1 B6 C5 D2 Do 18 A5 BC 45 BD Fo 15 A5
255o BC Do o2 C6 BD C6 BC 2o 1E 23 A5 CB 85 B6 2o 41
256o 21 4C 1A 24 4C 55 23 2o D1 21 2o F8 2o Bo oF 2o
257o 13 23 A1 Bo 2o 1C 27 2o 19 21 9o F6 Bo 2F 2o o8
258o 22 9o 33 2o AF 21 2o A2 21 Bo o3 2o 6D 21 2o E3
259o 21 Do oB Ao o2 B9 FC 27 2o 25 27 88 1o F7 A1 B8
25Ao C9 FF Fo o9 48 2o 1C 27 68 C9 oD Do DE 2o DA 21
25Bo 2o 4E 27 4C 5E 25 24 Co 1o 19 2o 12 22 2o E3 21
25Co Fo EB A1 B8 2o 1C 27 2o 6D 21 Bo E1 2o E3 21 Do
25Do F1 Fo DA 2o 12 22 2o C2 21 4C C2 25 2o 6C 2o 2o
25Eo 2B 2o 4C 12 2o 38 A5 Bo E9 o1 85 B4 A5 B1 E9 oo
25Fo 85 B5 2o 41 21 A1 B6 C9 o2 Do 3o 2o 22 2o 2o 8A
26oo 27 2o 1B 2o A1 Do C9 FF Fo o5 2o 9C 27 9o F2 2o
261o 6C 2o 2o 8A 27 A1 Do C9 FF Fo oA 81 Bo 2o 9C 27
262o 2o 19 21 9o Fo 2o 2B 2o 4C 53 26 2o 57 21 2o 22
263o 2o 2o 1B 2o 2o 41 21 A1 B6 C5 D2 Do F4 2o 6C 2o
264o 2o 2B 2o 2o 41 21 A1 B6 C5 D2 Fo o7 81 Bo 2o 19
265o 21 9o Fo 2o 12 2o 4C 1A 24 2o 41 21 2o D1 21 Ao
266o oo 84 C8 B1 B6 C5 D2 Fo 1A C1 Bo Fo oA 2o DA 21
267o 2o 19 21 Bo 1E 9o E5 A9 o1 85 C8 C8 2o 19 21 Bo
268o 12 9o Eo A5 C8 Fo oC 18 98 65 B6 85 B6 8A 65 B7
269o 85 B7 6o Ao oE B9 A1 26 2o 25 27 88 1o F7 4C 47
26Ao 23 21 6E 65 64 6E 75 66 65 67 2o 74 68 63 69 4E
26Bo 2o 59 26 84 C8 38 A5 Bo E5 C8 85 B4 A5 B1 E9 oo
26Co 85 B5 2o 22 2o 2o DC 25 2o 41 21 A1 B6 C5 D2 Do
26Do o3 4C 55 23 2o 57 21 4C E5 25 2o 3D 27 2o 13 23
26Eo 4C 5E 25 2o F8 2o 9o F2 2o 7F 27 2o 95 27 2o 22
26Fo 2o A5 B8 85 B4 A5 B9 85 B5 2o DC 25 4C 5E 25 2o
27oo 2E 27 2o 34 2o 4C EB 26 2o 8A 27 A1 Do C9 FF Do
271o o3 4C 5E 25 2o 1C 27 2o 9C 27 9o EF C9 oD Do o5
272o 48 2o 53 27 68 2o 58 27 2o Ao 1E 4C 5B 27 8A 85
273o C9 24 Co 3o o7 2o o8 22 Bo o2 E6 C9 6o 2o o6 23
274o A9 FF 81 B2 6o 2o 58 27 2o 5A 1E 4C 5B 27 A9 oD
275o 2o 25 27 A9 oA 4C 25 27 84 CD 6o A4 CD A2 oo 6o
276o 48 4A 4A 4A 4A 2o 6B 27 68 29 oF o9 3o C9 3A 9o
277o o2 69 o6 4C 25 27 A5 CE 85 Co A5 CF 85 C1 6o 2o
278o 2E 27 2o 12 22 9o F7 4C 6D 21 AD oF 2o 85 Do AD
279o oE 2o 85 D1 6o 46 C9 9o FB 4C C2 21 38 A5 Do ED
27Ao 11 2o A5 D1 ED 1o 2o Bo o7 E6 Do Do o2 E6 D1 6o
27Bo 2o B9 27 4C 56 2o 2o 8A 27 A9 FF 81 Do 6o 2o 8A
27Co 27 A1 Do C9 FF Fo F6 2o 9C 27 4C C1 27 2o D1 21
27Do 2o BE 27 2o 7F 27 2o 95 27 2o E3 21 Fo oE A1 B8
27Eo 81 Do 2o 9C 27 2o 6D 21 Bo o2 9o ED 2o B9 27 4C
27Fo DA 21 A9 5E 2o 25 27 A9 42 4C 25 27 29 5E 28 Co
```

Komplettes Editor-Programm als Hex-Dump. Die Adressenbelegung entspricht dem KIM-1, läßt sich jedoch (solange man keinen Transfer in andere Speicherbereiche vornimmt) sehr leicht an andere Systeme anpassen

Beispiel: !J5XCZEILE$LINE$T$$
1.LINE(^)
2.LINE(^)
3.LINE(^)
4.LINE(^)
5.LINE(^)

Nach dem C-Befehl steht der Zeiger also immer hinter dem geänderten Text. TEXT1 darf max. 255 Zeichen lang sein. TEXT1 und TEXT2 dürfen auch Carriage-Returns enthalten. Der C-Befehl, allein angewendet, gestattet es auch, einzelne Zeichenketten zu suchen und zu löschen, indem man CTEXT$$ eingibt. Wird TEXT1 nicht gefunden, verhält sich der KIM-Editor wie bei einem erfolglosen SEARCH.

Der Hilfsspeicher

Der Hilfsspeicher in Zusammenhang mit den B-Befehlen erlaubt dem Benutzer des KIM-Editors, Texte umzubauen.

BK (BUFFER/KILL)
löscht den Inhalt des Hilfsspeichers. Beim Kaltstart des Editors wird er automatisch gelöscht.

BR (BUFFER/READ)
gibt den gesamten Inhalt des Hilfsspeichers (zur Kontrolle) aus, ohne daß die Stellung des Zeigers im Hauptspeicher verändert wird.

nBC (BUFFER/COPY)
kopiert n Zeilen vom Zeiger an in den Hilfsspeicher. Die Stellung des Zeigers wird nicht verändert, das Kopierte wird an den bereits vorhandenen Inhalt im Hilfsspeicher angehängt.
Beispiel: !J3X1BC$$
!BR$$
1.LINE
1.LINE
1.LINE
!

nBT (BUFFER/TRANSFER)
arbeitet wie BC, nur daß gleichzeitig die kopierten Zeilen im Hauptspeicher gelöscht werden.
Beispiel: (Umsortieren der 5 Zeilen im Hauptspeicher):
!BKZ5X-1BT$$
!BR$$
5.LINE
4.LINE
3.LINE
2.LINE
1.LINE
!

Man kann also auch rückwärts kopieren. Im Hilfsspeicher stehen jetzt die Zeilen in umgekehrter Reihenfolge, der Hauptspeicher ist leer. (Bei B-Befehlen kann das # nicht verwendet werden.)

Jetzt fehlt noch ein Befehl, um den Inhalt des Hilfsspeichers an beliebiger Stelle wieder in den Text des Hauptspeichers einzufügen.

I^B (INSERT/BUFFER)
Das ^B ist das Echo des KIM-Editors für CNTRL B. Dieser Befehl fügt den gesamten Inhalt des Hilfsspeichers vor dem Zeiger in den Hauptspeicher ein.
Beispiel: !I^B#T$$
5.LINE
4.LINE
3.LINE
2.LINE
1.LINE
!

Zum Schluß noch einige Sonderbefehle:

H (HOME)
läßt den KIM-Editor nach 1C00 springen (KIM-Monitor).

V
zeigt dezimal die Anzahl der Zeilen im Hauptspeicher.

N
zeigt dezimal die Anzahl der Zeichen im Hauptspeicher.

E
zeigt hexadezimal die letzte momentan belegte Adresse im Hauptspeicher. Diese Adresse ist immer mit FF, der End-Marke, belegt. Beim Wegspeichern des Textes auf Kassette muß dieses FF unbedingt mitgespeichert werden!

Alle Befehle können dem KIM-Editor auch in Kleinschreibung mitgeteilt werden, was die Erstellung von Texten wesentlich erleichtert. Alle Zeichen, die nicht einem Befehl zugeordnet werden können, ignoriert der KIM-Editor, ebenso Zahlen vor Befehlen, die keine brauchen.

Zwei Besonderheiten des KIM-Editors bei Überschreiten von Speicherbereichen sollten noch erwähnt werden: Droht der Eingangsspeicher überzulaufen, dann schließt der KIM-Editor die Eingabe von sich aus zwangsweise mit zwei ESCAPES ab, die nicht geechot werden, und arbeitet den Eingangsspeicher ab. Droht der Hauptspeicher überzulaufen (bei I,C,I^B) oder aber der Hilfsspeicher (bei BC, BT), dann wird das letzte Einfügen von Text nicht mehr durchgeführt, es erfolgt die Fehlermeldung **Core** und der Zeiger wird an den Anfang des Hauptspeichers gestellt. Die weitere Abarbeitung von Befehlen wird abgebrochen.

Bei der Verwendung des X-Kommandos in Zusammenhang mit dem C-Befehl kann erhebliche Zeit bei der Abarbeitung der Befehle vergehen.

Beispiel: !200XIDER AFFE IST TOT
$$
braucht nur ca. 2 Sekunden;
!J200XCAFFE$IGEL$CIST
TOT$LEBT$$

dauert aber immerhin ca. 90 Sekunden! (Dieses Beispiel funktioniert nur mit größerem Hauptspeicher!)

Der Benutzer hat dafür zu sorgen, daß bei den angegebenen Bereichen tatsächlich RAM vorhanden ist, und daß die End-Adressen immer größer sind als die Anfangsadressen. Die Sprungadresse zum OUTPUT des KIM steht in 272A, 272B (A0,1E). Hier muß für einen TIM C6,72 eingesetzt werden. Die Sprungadresse zum INPUT des KIM steht in 274A,4B (5A,1E). Hier muß für einen TIM E9,72 eingesetzt werden.

Bei einigen Datensichtgeräten ist ESCAPE (1B) der Taste CNTRL SHFT K zugeordnet, was das Arbeiten mit dem KIM-Editor etwas mühsam macht. Wünscht der Benutzer einen anderen Delimiter, so braucht er nur das 1B in 234F durch den Wert eines anderen Control-Zeichens zu ersetzen. Die Werte 02, 12, 18 und 7F sind allerdings nicht erlaubt, da sie schon belegt sind.

Soll der KIM-Editor bei dem Befehl H nicht nach 1C00 springen, so sind die Werte in 246C (00) und 246D (1C) entsprechend zu ändern.

Adressenanpassung

Werden vom Benutzer andere Speicherbereiche für Eingangs-, Haupt- und Hilfsspeicher gewünscht, so ist eine Änderung leicht möglich, denn diese Adressen stehen im Programm nur an den Stellen 2006-2011, und zwar in der Reihenfolge:

Eingangsspeicher-Anfang,	H-Byte: 2006
Eingangsspeicher-Anfang,	L-Byte: 2007
Eingangsspeicher-Ende,	H-Byte: 2008
Eingangsspeicher-Ende,	L-Byte: 2009
Hauptspeicher-Anfang,	H-Byte: 200A
Hauptspeicher-Anfang,	L-Byte: 200B
Hauptspeicher-Ende,	H-Byte: 200C
Hauptspeicher-Ende,	L-Byte: 200D
Hilfsspeicher-Anfang,	H-Byte: 200E
Hilfsspeicher-Anfang,	L-Byte: 200F
Hilfsspeicher-Ende,	H-Byte: 2010
Hilfsspeicher-Ende,	L-Byte: 2011

Befehlsliste

(Weglassen von n entspricht 0n; n immer dezimal angeben)

nX	Die folgenden Befehle n mal ausführen
V	Ausgabe der Zeilenzahl im Speicher
N	Ausgabe der Zeichenzahl im Speicher
E	Momentane Endadresse des Hauptspeichers hexadezimal angeben
T	Momentane Zeile mit Zeiger (^) ausgeben
nT	n Zeilen ausgeben (# für n: alles ausgeben)
nL	Zeiger um n Zeilen vorrücken
0L	Zeiger an den Anfang der Zeile
nM	Zeiger um n Zeichen vorrücken
nD	n Zeichen löschen
nK	n Zeilen löschen (# für n: alles löschen)
0K	Zeile vom Zeiger an rückwärts löschen
I	Text einfügen
^R	Laufende Zeile im Eingangsspeicher zeigen
^X	Eingabe löschen
J	Zeiger zum Beginn des Hauptspeicher-Textes
Z	Zeiger an das Textende im Hauptspeicher
H	Rückkehr ins Monitorprogramm
S	Zeichenkette suchen
C	Zeichenkette ändern
DEL	Letztes Zeichen löschen und wieder ausgeben
nBC	n Zeilen vom Zeiger an in den Hilfsspeicher kopieren
nBT	Wie BC, aber Zeilen im Hauptspeicher löschen
BK	Hilfsspeicher löschen
BR	Inhalt des Hilfsspeichers ausgeben
I^B	Inhalt des Hilfsspeichers einfügen

Zeilen mit „Return" abschließen. Mehrere Befehle können als Kette eingegeben werden, Ausführung nach $$.
Fehlermeldungen: „Nicht gefunden" bei S und C; „+++ CORE +++" wenn kein Speicherplatz mehr frei ist. Der Zeiger steht danach stets am Textanfang. Alle Befehle dürfen auch in Kleinschreibung eingegeben werden. ■

HP-41-Tips

Unter bestimmten Umständen kann der HP-41-Benutzer vor dem Problem stehen, daß sich dieser programmierbare Taschenrechner intern „aufgehängt" hat und sich von der Tastatur aus nicht mehr bedienen läßt. Durch kurzes Entfernen der Batterien kann der Rechner auch ohne „Memory Lost" zurückgesetzt werden. (Die Batterien halten übrigens bei häufigem Gebrauch der Akustik-Funktionen knapp zwei Monate.)

Ist im X-Stack ein Doppelpunkt, Komma oder Punkt gespeichert und das Alpha-Register mit über 11 Zeichen gefüllt, so „erscheint" das letzte Zeichen im Alpharegister an erster Stelle im X-Register, wenn man vom Alpha- in den Normalmodus per Tastendruck umschaltet. Nach zweimaligem x-y-Tausch ist der Spuk wieder vorbei.

Um z.B. Register 10 mit Register 15 zu vertauschen, kann man folgende Befehlsfolge verwenden: x ‹› 10 / x ‹› 15 / x ‹› 10. Ebenso ist die Folge x ‹› 15 / x ‹› 10 / x ‹› 15 geeignet. Der Stack wird dabei nicht verändert.

Beim Programmieren mit UPN ist es sehr hilfreich, wenn man die Befehle waagrecht nebeneinander und die zugehörigen Stackpositionen (X,Y,Z,T) senkrecht darüber schreibt. Das Verfahren läßt sich natürlich auch auf Last X erweitern und gibt einen genauen Einblick in die UPN-Logik.

Peter Obermeier

Der Nascom-1 besitzt von Haus aus ein ausgefeiltes Betriebssystem. Allerdings wünscht man sich in der Praxis noch einige Leistungen, die das Programmieren weiter vereinfachen.

Dipl.-Ing. Gustav Wostrack

Erweitertes Betriebssystem für den Nascom-1

Das Betriebssystem für den Nascom-1 kann um einige Befehle erweitert werden. Wird das Programm (*Bild 1*) bei 0C60 gestartet, so stehen weiterhin zunächst alle bekannten Befehle zur Verfügung. Darüber hinaus sind nun folgende Kommandos implementiert:

A (ASCII): Eingabe und Änderung von ASCII-Zeichen
W (write): Auflistung des Programmspeichers in besonderem Format
+ : Addition zweier HEX-Zahlen
– : Subtraktion zweier HEX-Zahlen
P (print): Ausgabe von Baudot-Zeichen
V (Video): Ausgabe auf dem Bildschirm
c (copy2): Verschieben von Speicherstellen

Ferner besteht nun die Möglichkeit, auch *kleine* Buchstaben auf dem Bildschirm darzustellen. Dabei ist zu beachten, daß die Shift-Taste die genau umgekehrte Funktion wie bei einer Schreibmaschine besitzt: Shift-Taste + Buchstabentaste erzeugt einen Kleinbuchstaben. Bei der Eingabe des Programmes muß noch berücksichtigt werden, daß die Speicherstellen 0CBC bis 0CF9 nicht eingegeben werden brauchen, da diese automatisch aus dem vorhandenen Monitorprogramm kopiert werden.

Beschreibung der Kommandos

A (ASCII)

Mit Hilfe dieses Kommandos können ASCII-Zeichen direkt in den Speicher geschrieben werden. Dabei ist es nicht mehr nötig, den sedezimalen Wert eines Zeichens nachzuschlagen (z. B. A = 41) und zweistellig einzugeben, sondern es wird lediglich die dem gewünschten Zeichen entsprechende Taste des Keyboard gedrückt. Dabei ist die Tatsache, daß nun auch Kleinbuchstaben verfügbar sind, von großem Nutzen. Die Handhabung dieses Kommandos erfolgt in ähnlicher Weise wie die des Befehls M (modify). Lediglich werden die Steuerzeichen in einer etwas anderen Form benutzt (siehe dazu auch das Beispiel in *Bild 2*).

Format: Axxxx
xxxx: Adresse der Speicherstelle, an der das erste Zeichen eingeschrieben oder geändert werden soll.

Bedeutung der Steuer- und Sonderzeichen

Alle Eingaben sind durch New Line abzuschließen. Dabei ist allerdings die Bedeutung der Zeichen @ und Shift 0 zu berücksichtigen:

Taste @	keine Veränderung, nächstes Zeichen	
Taste Shift und Taste a:	Speicherung von EF_{16} (RST 5)	
Taste Shift und Taste 0:	Speicherung von 00_{16} und Rücksprung in das Betriebssystem	
Taste BS:	Speicherung von $1D_{16}$	beim Drücken der entsprechenden Taste direkt nach Prompt
Taste FF:	Speicherung von $1E_{16}$	
Taste NL:	Speicherung von $1F_{16}$	
Taste . (Punkt):	Beendigung der Eingabe; nur nach Eingabe eines Zeichens möglich.	

W (write)

Dieser Befehl dient hauptsächlich der schriftlichen Dokumentation von Programmen. Denn zur Ausgabe von Hex-Dumps ist das Kommando T (tabulate) schon allein wegen des ungünstigen Formats ungeeignet. Der Ausdruck in Bild 1 ist mit dem Kommando W erstellt worden. Die herausragenden Merkmale sind:
– automatische Erstellung eines Kopfes
– automatische Seitennumerierung
– optimale Ausnutzung der Papierbreite
– der Ausdruck wird auf DIN A4 formatiert; dies ist insbesondere bei längeren Ausdrucken hilfreich
– der Ausdruck endet genau mit dem Speicherinhalt, der gewünscht wird; nicht erst nach dem Ausdruck einer vollen Achterreihe.

Format: Wxxxx yyyy
xxxx: Adresse der ersten Speicherzelle, die ausgedruckt werden soll
yyyy: Adresse der letzten Speicherzelle, die ausgedruckt werden soll.

Nach Ausdruck des Wortes „Programm“: (auf dem Bildschirm erscheint lediglich das Zeichen £; die Erklärung dazu folgt unter der Beschreibung des Kommandos P) kann ein Programmname eingegeben werden, der mit New Line abzuschließen ist.

+ und –

Oftmals besteht bei der Programmerstellung die Notwendigkeit, sedezimale Zahlen (Hex-Zahlen) zu addieren oder zu subtrahieren, beispielsweise bei der Berechnung der Distanzadresse relativer Sprungbefehle.

Format: +xxxx yyyy
=zzzz
xxxx: 1. Summand
yyyy: 2. Summand
zzzz: Summe

–xxxx yyyy
=zzzz
xxxx: Subtrahend
yyyy: Minuend
zzzz: Differenz

P (print)

Bei der Benutzung eines Mikrocomputers erwächst sehr bald der Wunsch, Programme oder auch Programmausdrucke schriftlich dokumentieren zu können. Aus finanziellen Gründen liegt es nahe, sich für einen meist kostengünstigen Fernschreiber als Peripheriegerät zu entscheiden.

Die Schwierigkeiten, die dabei auftreten, liegen meist weniger in der

```
nascom 1

programm:   erweitertes betriebssystem fuer den nascom 1

page 01

0c60 21 7f 0c 22 45 0c 21 b6 0c 11 3f 0c 01 06 00 ed
0c70 b0 21 ea 00 11 bc 0c 01 3e 00 ed b0 c3 59 03 4d
0c80 ad 01 43 ef 03 45 d0 02 53 ff 02 54 fc 01 42 e3
0c90 00 4c 7c 03 44 d1 03 41 3b 0d 50 2a 0e 56 c3 0d
0ca0 2b 1a 0d 2d 20 0d 57 cf 0d 63 3f 0f 00 00 00 00
0cb0 00 00 00 00 00 00 5e 00 1d 00 bc 0c 08 88 09 14
0cc0 9c 9b a3 92 c2 ba b2 aa a2 98 a0 29 0a 21 19 1a
0cd0 1c 1b 23 12 42 3a 32 2a 22 18 20 a9 8a a1 99 0d
0ce0 2c 41 13 3b 33 43 10 40 2d 38 30 28 31 39 25 1d
0cf0 24 15 34 45 35 11 2b 44 3d 3c 9a 00 8d 89 00 00
0d00 ac c1 93 bb b3 c3 90 c0 ad b8 b0 a8 b1 b9 a5 9d
0d10 a4 95 b4 c5 b5 91 ab c4 bd bc cd 33 0d 19 18 06
0d20 cd 33 0d a7 ed 52 ef 20 20 20 3d 00 cd 32 02 ef
0d30 1f 00 c9 2a 0c 0c ed 5b 0e 0c c9 2a 0c 0c cd 32
0d40 02 7e cd 8f 0d ef 3e 00 cd 3e 00 fe 40 28 28 fe
0d50 5b 20 09 97 77 ef 30 30 2e 1f 00 c9 fe 5d 20 02
0d60 3e ef 77 cd 8f 0d cd 3e 00 fe 1d 20 06 ef 1d 1d
0d70 00 18 d5 fe 1f 20 06 ef 1f 00 23 18 c1 fe 2e 20
0d80 e5 cd 4a 0c cd 3e 00 fe 1f 20 f9 cd 4a 0c c9 fe
0d90 1d 20 05 ef 42 53 00 c9 fe 1e 20 05 ef 46 46 00
0da0 c9 fe 1f 20 05 ef 43 52 00 c9 fe 00 20 05 ef 30
0db0 30 00 c9 fe ef 20 05 ef 50 52 00 c9 cd 4a 0c ef
0dc0 20 00 c9 21 3b 01 22 4b 0c 3e 00 32 00 0c c9 cd
0dd0 33 0d 97 08 ef 1f 1f 1f 1f 1f 40 1f 1f 23 20 20
0de0 20 00 cd de 01 ef 1f 50 61 67 65 20 00 08 3c f5
0df0 cd 44 02 f1 08 ef 1f 1f 00 0e 20 06 10 d5 c5 cd
0e00 32 02 c1 d1 7e cd 44 02 cd 3c 02 e5 d5 a7 eb ed
0e10 52 d1 e1 ca 2f 0d 23 05 20 ea ef 1f 00 0d 20 db
0e20 ef 1f 1f 1f 1f 2d 2d 00 18 aa 21 3c 0e 22 4b 0c
0e30 3e fb 32 50 0c cd 5d 00 3e 10 18 8f cd 3b 01 c5
0e40 e5 d5 fe 40 f5 3a 50 0c 4f 30 0d 3e fb b9 28 1b
0e50 32 50 0c cd 5d 00 18 13 f1 fe 60 38 02 d6 20 f5
0e60 3e ff b9 28 06 32 50 0c cd 5d 00 f1 21 89 0e 01
0e70 5f 00 11 5a 00 ed b1 e2 85 0e e5 f5 19 7e cd 5d
0e80 00 f1 e1 18 f0 d1 e1 c1 c9 07 20 22 27 28 29 2b
0e90 2c 2d 2e 2f 30 31 32 33 34 35 36 37 38 39 3a 3b
0ea0 3c 3d 3e 3f 41 42 43 44 45 46 47 48 49 4a 4b 4c
0eb0 4d 4e 4f 50 51 52 53 54 55 56 57 58 59 5a 1f 1f
0ec0 40 40 40 40 40 40 40 40 40 40 23 23 23 23 23 23
0ed0 23 23 23 23 23 24 24 24 24 24 24 24 24 24 24 24
0ee0 24 24 24 24 eb e4 e5 e5 ef f2 f1 ec e3 fc fd f6
0ef0 f7 f3 e1 ea f0 f5 e7 e6 f8 ee ec ef fe f2 f9 e3
0f00 f9 ee e9 e1 ed fa f4 e6 eb ef f2 fc ec f8 f6 f7
0f10 ea e5 f0 e7 fe f3 fd f5 f1 e8 e2 ec e3 e5 ee f8
0f20 fc e4 fb f7 ff ff f6 ea f8 fa ea e3 fc fc fb ee
0f30 ff fa fb fc ff e4 f3 f8 e5 f0 ea e3 ee ef fb cd
0f40 33 0d ed 4b 10 0c ed b8 c9
```

Bild 1. Vollständiges Hex-Listing der Betriebssystem-Erweiterung für den Mikrocomputer Nascom-1

Schnittstelle, die oft mit dem Computer mitgeliefert wird, als vielmehr in der Softwareanpassung. Dabei gibt es dann im wesentlichen zwei Probleme zu lösen:

1. Die Umsetzung des Codes ASCII in Baudot und
2. 8 bit Wortlänge im Computer und 5 bit Wortlänge im Fernschreiber zu vereinen.

Punkt 2 läßt sich ganz leicht lösen: Man füllt lediglich die 3 benötigten Bits mit dem Wert 111 auf, die dann vom Fernschreiber als Stopbits erkannt werden. Allerdings verringert sich dadurch die Übertragungsgeschwindigkeit geringfügig. Der Ausdruck einer DIN-A4-Seite mit dem Kommando W dauert etwa neun Minuten.

Das Problem Nr. 1 gestaltet sich schon als etwas schwieriger. Weil der Fernschreiber die Information benötigt, ob es sich bei dem übertragenen Zeichen um einen Buchstaben oder eine Zahl handelt, muß diese Frage zunächst durch Software geklärt und dann dem Fernschreiber mitgeteilt werden. Das vorliegende Programm weist darüber hinaus noch eine Besonderheit auf. Beim Umsetzvorgang der beiden Codes wird eine Liste durchsucht, die alle die Zeichen enthält, die übersetzt werden können. Wurde das Zeichen gefunden, so wird über eine Distanzadresse das entsprechende Baudotzeichen aus einer zweiten Liste entnommen und übertragen. Daraufhin wird die erste Liste weiterhin durchsucht. Wird dabei das gleiche ASCII-Zeichen noch einmal gefunden, so wird auch das hierzu entsprechende Baudotzeichen übertragen. Dieser Vorgang wiederholt sich, bis das Listenende erreicht ist. Dadurch wird es möglich, durch Drücken einer einzigen

```
)
)              0c5b ' )cr
)              0c5c cr)00.
)              )
)beispiele     )m0c5d
)              0c5d 39)c3 59 03.
)a0c51         )
0c51 4 )pr     )e0c51
0c52 v )f      funkschau
0c53 pr)u      )
0c54 2 )n      )+1234 9876
0c55 ß )k          =aaaa
0c56 , )s      )
0c57 - )c      )-aaaa 1234
0c58 j )h          =9876
0c59 ff)a      )
0c5a ß )u      )v
```

Bild 2. Einige Beispiele für den Umgang mit den Erweiterungs-Kommandos

Taste einen ganzen Schriftzug auf dem Fernschreiber ausdrucken zu lassen.

Allerdings wird auf dem Bildschirm nur einmal das ASCII-Zeichen erscheinen. Beispielsweise ist das Zeichen @ (40_{16}) in der ersten Liste 10mal aufgeführt. Also wird beim Drücken dieser Taste auch ein 10stelliger Ausdruck erfolgen, nämlich: nascom 1 (Bu und Zi sind mitzuzählen); ebenso bei $ G. WOSTRACK (Name des Verfassers) und £ PROGRAMM.

Erweitert man die schon vorhandene Liste, so ist dabei zu beachten, daß die Längenangabe der Liste und die Distanzadresse in Speicherzelle 0E70/71 und 0E73/74 entsprechend zu ändern sind.

Bei Aufruf des Kommandos P wird die LED eingeschaltet, die auch bei Ausführung der Kommandos D und L leuchtet. Diese Schaltfunktion kann beispielsweise dazu benutzt werden, die Fernschreibmaschine zu aktivieren.

V (Video)

Das Kommando V neutralisiert den Befehl P, so daß die Ausgabe ausschließlich auf dem Bildschirm erfolgt. Auch die oben beschriebene LED wird gelöscht.

c (copy2)

Oftmals ist es nötig, während der Debugging-Phase in ein Programm ein oder mehrere Byte einzufügen. Dies ist mit dem bekannten Befehl C (Copy) nur sehr schlecht möglich. Dazu wurde das neue Kommando c (copy2) installiert, das diese Tätigkeit sehr vereinfacht.
Format:

cxxxx yyyy zzzz
- xxxx: alte Endadresse
- yyyy: neue Endadresse
- zzzz: Distanzadresse, d. h. Differenz aus neuer Endadresse und Adresse, hinter der eingefügt werden soll. ■

AIM-Directory

Die Mikrocomputer AIM-65 und PC-100 benutzen zum Aufzeichnen von Maschinenprogrammen ein blockweises Format, das es gestattet, in einer einzigen Aufzeichnung (File) mehrere unterschiedliche Adressenbereiche anzusprechen. Leider kompliziert dies etwas das spätere Herausfinden jener Adressenbereiche – etwa, wenn man in einer Schublade eine verstaubte Kassette mit einem alten AIM-Programm findet, für das keine Dokumentation auffindbar ist.

```
010F          PASS 2
0000
0000          ;TAPE DIRECTORY
0000          ;BY H.FEICHTINGER
0000 CNT              =$FB
0000 SVX              =$FC
0000 NEWL             =$FD
0000 NEWH             =$FE
0000 EQFLG            =$FF
0000 NXTAD            =$E2CD
0000 RBYTE            =$E3FD
0000 WHERI            =$E848
0000 NUMA             =$EA46
0000 OUTAL            =$E9BC
0000 INALL            =$E993
0000 CRLF             =$E9F0
0000 DU13             =$E520
0000 MONI             =$E182
0000 ADDR             =$A41C
0000                  *=$E00

0E00 START 2048E8     JSR WHERI
0E03       A900       LDA #0
0E05       85FB       STA CNT
0E07       A9FD       LDA #$FD
0E09       8D1CA4     STA ADDR
0E0C       8D1DA4     STA ADDR+1
0E0F NWBL  85FF       STA EQFLG
0E11 SRCH  2093E9     JSR INALL
0E14       C93B       CMP #';
0E16       D0F9       BNE SRCH
0E18       20FDE3     JSR RBYTE
0E1B       AA         TAX
0E1C       20FDE3     JSR RBYTE
0E1F       CD1DA4     CMP ADDR+1
0E22       85FE       STA NEWH
0E24       F002       BEQ *+4
0E26       E6FF       INC EQFLG
0E28       20FDE3     JSR RBYTE
0E2B       CD1CA4     CMP ADDR
0E2E       85FD       STA NEWL
0E30       F002       BEQ *+4
0E32       E6FF       INC EQFLG
0E34       A5FF       LDA EQFLG
0E36       F021       BEQ DATA
0E38       300C       BMI NWAD
0E3A       AD1DA4     LDA ADDR+1
0E3D       20950E     JSR HEX
0E40       AD1CA4     LDA ADDR
0E43       20950E     JSR HEX
0E46 NWAD  8A         TXA
0E47       F025       BEQ RDY
0E49       A5FE       LDA NEWH
0E4B       8D1DA4     STA ADDR+1
0E4E       20950E     JSR HEX
0E51       A5FD       LDA NEWL
0E53       8D1CA4     STA ADDR
0E56       20950E     JSR HEX
0E59 DATA  20FDE3     JSR RBYTE
0E5C       A001       LDY #1
0E5E       20CDE2     JSR NXTAD
0E61       CA         DEX
0E62       D0F5       BNE DATA
0E64       20FDE3     JSR RBYTE
0E67       20FDE3     JSR RBYTE
0E6A       A900       LDA #0
0E6C       F0A1       BEQ NWBL
0E6E RDY   2023E5     JSR DU13+3
0E71       A200       LDX #0
0E73       F003       BEQ *+5
0E75 NXT   20F0E9     JSR CRLF
0E78       A920       LDA #$20
0E7A       20BCE9     JSR OUTAL
0E7D       20A10E     JSR BFOUT
0E80       20A10E     JSR BFOUT
0E83       A92D       LDA #'-
0E85       20BCE9     JSR OUTAL
0E88       20A10E     JSR BFOUT
0E8B       20A10E     JSR BFOUT
0E8E       E4FB       CPX CNT
0E90       90E3       BCC NXT
0E92       4C82E1     JMP MONI
0E95 HEX   86FC       STX SVX
0E97       A6FB       LDX CNT
0E99       9DA80E     STA BUF,X
0E9C       E6FB       INC CNT
0E9E       A6FC       LDX SVX
0EA0       60         RTS
0EA1 BFOUT BDA80E     LDA BUF,X
0EA4       E8         INX
0EA5       4C46EA     JMP NUMA
0EA8 BUF
0EA8                  *=$10C
010C       4C000E     JMP START
010F                  .END
010F        ERRORS= 0000
```

Bild 1. Programm zum Auflisten von auf Kassette gespeicherten Adressenbereichen. Zur Eingabe verwendet man am besten die Taste „/" und tippt nur die hexadezimalen Bytes ein, also 2048 E8 A9 usw. ab Adresse 0E00. Der Sprung bei 010C gestattet den Programmstart mit Taste F1

Das in *Bild 1* wiedergegebene Directory-Programm erstellt einen Ausdruck aller innerhalb einer File abgespeicherten Adressenbereiche. Startet man es bei 0E00, so gibt man wie sonst beim Laden nach „IN=" einfach T ein und nach „F=" den Filenamen (er läßt sich leicht durch probeweises Laden mit einem ungültigen Filenamen, z.B. XYZ, herausfinden; der AIM zeigt dann SRCH und den tatsächlichen Namen an). Nach „T=" kann man z.B. zweimal Return drücken. In der Anzeige erscheint dann „LOAD", und sobald die gesamte File durchsucht wurde – sie wird hier nicht wirklich geladen – so wird auf dem Drucker eine Liste aller enthaltenen Adressenbereiche ausgegeben, vorausgesetzt, dieser wurde vorher eingeschaltet (*Bild 2*).

Bild 2. Hier wurde ein Programm namens „RTTY" analysiert

```
<G>/
IN=T F=RTTY T=1
 0000-00A9
 0200-0301
 0170-01A9
```

Zu beachten ist, daß die ausgedruckten Endadressen im Gegensatz zu den bei der Aufnahme angegebenen nicht das letzte Byte der Aufzeichnung, sondern das darauf folgende, also nicht mehr enthaltene bezeichnen. Wurde z.B. der Bereich 0200...02FF aufgezeichnet, so würde nun 0200-0300 ausgedruckt.

Das Programm benutzt den Bereich ab 0EA8, um die auszugebenden Adressen zwischenzuspeichern, bis der Ladevorgang abgeschlossen ist, sowie den Zero-Page-Bereich von 00FB bis 00FF.

Herwig Feichtinger

Ein drahtloser Plausch mit Australien, technische Gespräche mit Gleichgesinnten in den USA – diese und andere Wünsche wecken in zahlreichen Funkbegeisterten den Gedanken an eine eigene Kurzwellen-Sendestation. Ein Anwärter auf die begehrte Lizenzurkunde muß bei der Bundespost Kenntnisse über Technik und Gesetze nachweisen – und er muß morsen können.

Erich H. Franke

KIM als Morse-Trainer

Nun, alles läßt sich erlernen. Der Kandidat findet z. B. beim DARC, einer deutschen Amateurfunkvereinigung, Hilfe, unter anderem in Form eines Morsekurses auf Kassetten. Dieser enthält Morsetexte, die mitgeschrieben und mit der „Auflösung" im Begleitheft verglichen werden können. Die Texte bestehen teils aus Klarschrift, teils aber auch aus Gruppen zu je fünf Zeichen, die zufällig ausgewählt sind, um reines Auswendiglernen zu erschweren. Immer wieder kommt es jedoch vor, daß an Schulungsabenden in den Ortsverbänden die Teilnehmer klagen, sie hätten nach wie vor Probleme beim Mitschreiben der Zeichen, könnten die Übungstexte aber längst auswendig.

Das vorliegende Programm für den KIM-1 soll hier Abhilfe schaffen. Es erzeugt zufällige Fünfergruppen als Nf-Signal und protokolliert diese auf einem angeschlossenen Fernschreiber. Dabei können Zeichenmenge, Ausgabegeschwindigkeit und Zeichenabstand den Kenntnissen des Lernenden angepaßt werden. Es stehen ihm dazu elf Lektionen zur Verfügung. In der ersten werden nur vier, in der letzten alle 44 für die Prüfung notwendigen Zeichen in die Gruppen aufgenommen. Hilfen wie doppelter Wortabstand und stufenweise Steigerung der Geschwindigkeit sind ebenfalls vorhanden.

Das Programm läuft vollständig im KIM-1-System. Speichererweiterungen und Umbauten an der Platine sind nicht nötig. Da sowohl die Töne als auch die Fernschreibzeichen vom Prozessor erzeugt werden, genügt als Ausgabegerät die Schaltung nach *Bild 1*. Der Port PA 0 (Anschluß A-14 auf dem Applikationsstecker) dient als Ausgang für die Morsezeichen. Sie werden einem Kassettenrecorder zwecks Aufzeichnung zugeführt und gleichzeitig über den Lautsprecher hörbar gemacht. Das Fernschreib-Interface wird von PB 0 (Anschluß A-9) gesteuert. Eine alternative Schaltung findet sich in FUNKSCHAU 1979, Heft 1.

Das Programm selbst (*Bild 2*) besteht aus zwei Blöcken. Es belegt mit allen für den Betrieb notwendigen Tabellen die

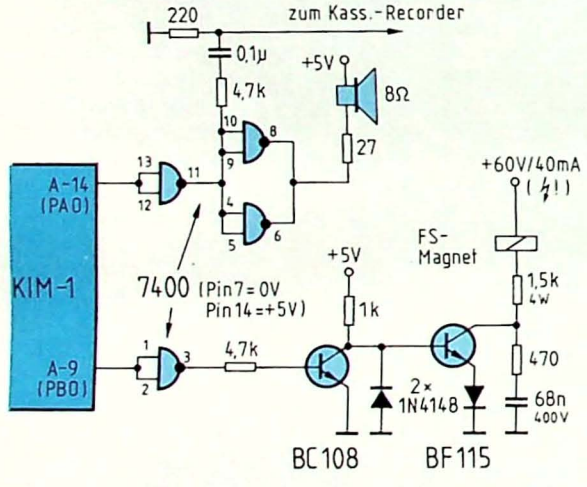


Bild 1. Einfache Interface-Schaltung für Lautsprecher, Kassettenrecorder und Fernschreiber

```
0000 40 80 03 05 2C 21 06 00 1B 02 FF 30 0E 05 42 42
0010 41 CA 2B 21 28 0D 45 4C 30 56 51 53 41 3D 54 32
0020 4F 43 44 35 52 2F 49 39 47 58 46 34 3F 54 4E 4A
0030 55 37 48 2C 4B 42 33 50 4D 59 5A 57 36 31 38 2E
0040 28 29 40 80 03 05 2C 21 06 22 02 00 00 36 4A 16
0050 00 00 00 7A B4 B6 44 14 CE 86 56 94 FC 7C 3C 1C
0060 0C 04 84 C4 E4 F4 E2 AA AC 8C 54 32 00 60 88 A8
0070 90 40 28 D0 08 20 78 B0 48 E0 A0 F0 68 D8 50 10
0080 C0 30 18 70 98 B8 C8 00 00 00 00 00 11 94 DC 91

0200 A2 06 B5 42 95 00 CA 10 F9 A2 01 8E 03 17 8E 01
0210 17 8E 02 17 CA 8E 00 17 A5 4A 48 A5 49 48 08 4C
0220 00 1C 20 DA 02 20 DA 02 A9 3C 20 73 02 A6 03 20
0230 92 02 A2 0F 86 0C A2 05 86 0D 38 A5 0F 65 12 65
0240 13 85 0E A2 04 B5 0E 95 0F CA 10 F9 29 3F C5 04
0250 B0 E8 AA B5 16 20 E1 02 20 73 02 C6 0D D0 DB A6
0260 03 20 92 02 20 DA 02 C6 0C D0 CB A9 3E 20 73 02
0270 4C 18 02 A8 B9 2C 00 38 F0 01 18 A0 02 A2 01 2A
0280 90 02 A2 03 20 94 02 A2 01 20 92 02 C9 80 D0 EA
0290 A6 02 A0 00 48 98 A4 01 84 14 8D 00 17 A4 00 8C
02A0 05 17 AC 05 17 10 FB C9 00 F0 02 49 01 C6 14 D0
02B0 E9 CA D0 E2 8E 00 17 68 60 A5 06 85 0A A5 05 38
02C0 E9 01 B0 02 C6 0A A4 0A 10 F5 60 A5 06 85 0A A5
02D0 05 4A 46 0A 90 E9 09 80 B0 E6 A9 0D 20 E1 02 A9
02E0 0A 85 0B 20 1F 03 85 07 86 09 20 B9 02 AD 02 17
02F0 29 FE 8D 02 17 20 B9 02 A2 05 AD 02 17 29 FE 46

0300 07 69 00 8D 02 17 20 B9 02 CA D0 EE AD 02 17 09
0310 01 8D 02 17 20 B9 02 20 CB 02 A6 09 A5 0B 60 86
0320 15 AA BD 5F 03 29 C0 F0 23 10 0E 0A 10 1B 20 54
0330 03 A9 0D 20 E6 02 4C 4C 03 24 08 30 0F A9 80 85
0340 08 A9 1F 20 E6 02 4C 4C 03 20 54 03 BD 5F 03 29
0350 1F A6 15 60 24 08 10 FB A9 1B 85 08 4C E6 02 00
0360 08 00 00 00 89 00 0B 00 04 02 00 00 08 1F 1B 00
0370 00 00 00 00 1F 00 00 00 00 00 09 00 00 00 00 04
0380 9C 85 9B 5F 89 91 85 8F 92 91 91 8C 83 9C 9D 96
0390 97 93 81 8A 90 95 87 86 98 8E 8E 8F 9E 92 99 5F
03A0 43 59 4E 49 41 4D 5A 54 46 4B 4F 52 5C 4C 58 56
03B0 57 4A 45 50 47 5E 53 5D 55 51 8F 9D 92 8B 83 FE
```

Bild 2. ▶
Hexadezimale Auflistung des Morseübungs-Programms

Tabelle 1: Wichtige Variablen des Morse-Trainers

0000	Tonhöhe der Morsezeichen
0001	Zeichenlänge
0002	Zeichenabstand
0003	Wortabstand
0004	Nummer der Lektion (00...2C)
0005	BAUDL
0006	BAUDH

Tabelle 2: Einstellen der Fernschreiber-Geschwindigkeit

Baud	BAUDH	BAUDL
45,45	06	21
50	05	80
75	03	C7
100	02	C9

Speicherzellen 0016 bis 008B sowie 0200 bis 03BE. Die Startadresse ist 0200.

Die Bedienung ist weitgehend unkritisch. Nach dem Start erscheint auf der Anzeige „0222 20", die Adresse des „Warmstart-Punktes". In der Zero Page sind sieben Speicherplätze für Programmparameter reserviert, die Zeichenlänge usw. bestimmen. Diese kann man nun nach Belieben abändern, um das Programm seinen Bedürfnissen anzupassen. *Tabelle 1* zeigt die Adressen dieser Variablen und ihre Wirkungen. Der Wert in BAUDH und BAUDL bestimmt die Schrittgeschwindigkeit des abgegebenen Fernschreibsignales. Hierfür ist ein Standardwert von 45,45 Baud eingesetzt; dieser muß gegebenenfalls für vorhandene Geräte abgeändert werden. *Tabelle 2* gibt darüber Auskunft.

Ist nun alles nach Wunsch geändert, so kommt man durch Druck auf PC wieder zur Anzeige „0222 20" und kann das Programm nun endgültig mit „GO" in Gang setzen. Auf dem Fernschreiber wird nun zweimal Wagenrücklauf-Zeilenvorschub ausgegeben, als Morsezeichen erscheint „Verkehrsbeginn" (ct). Daraufhin folgen 15 Fünfergruppen, die zufällig aus dem Lektionsstoff ausgewählt werden und mit dem „Schlußzeichen" (+) enden. Das Programm hält nun wieder bei „0222 20" an und kann – mit oder ohne neuerliche Parameter-Änderungen – beliebig oft mit „GO" wieder gestartet werden.

Das alles funktioniert so: Nach der Initialisierung bei 0200 wird mit dem Zufallsgenerator eine Zahl zwischen 1 und einem von der Lektion abhängigen Maximalwert erzeugt. Mit diesem Zeiger wird aus der Tabelle bei 0016 ein ASCII-Zeichen geholt, das – in Baudot gewandelt – zum Fernschreiber geschickt und als Morsezeichen auf Kassette gespeichert wird.

Das Fernschreiber-Ausgabeprogramm bei 02E1 ist ein Modul, das häufig benötigt wird. Es kann über die Tabelle bei 035F auch Symbole ausgeben, die im Fernschreib-Code selbst nicht enthalten sind. Diese werden als Folge zweier Symbole dargestellt. Das Programm bei 02DA gibt einen Wagenrücklauf/Zeilenvorschub aus. Stolze Besitzer einer Teletype oder eines anderen Ausgabegerätes können an dieser Stelle auch Sprünge zu eigenen Ausgabegeräten einsetzen.

Dieser Trainer ist vor allem auch für Ortsverbände interessant, die selbst Morseübungen abhalten und diese sinnvoll ergänzen wollen, zumal auf dieser Basis die Beschaffung der notwendigen Geräte heutzutage kaum mehr ein Problem darstellt. ■

FSK-Demodulation per Software

Ein Modem besteht aus Modulator und Demodulator zur Übertragung von Daten auf einer Tonfrequenzleitung, z. B. per Telefon oder Funk. Meist verwendet man dazu ein Zwei-Frequenzen-Verfahren, auf Neuhochdeutsch Frequency Shift Keying (FSK). In FUNKSCHAU 14 und 15/1978 wurde ein solches Modem unter der Rubrik „Schaltungskniffe" beschrieben. Natürlich läßt sich die dafür erforderliche Hardware durch Software reduzieren.

Hier wird nun ein Demodulator beschrieben, der aus einem FSK-Signal (z. B. Kansas-City-Standard) ein Digitalsignal rückgewinnt, das entweder einem Terminal zugeleitet oder aber vom Mikrocomputer selbst weiterverarbeitet werden kann.

Der Demodulator arbeitet nach einem einfachen Algorithmus, der aus *Bild 1* hervorgeht. Nach dem Programmstart wartet er auf eine (positive oder negative) Flanke des Eingangssignales. Nach einer Verzögerungszeit von 3/4 der Periodenzeit der höheren Frequenz wird der Eingang erneut abgefragt. Hat sich sein Zustand geändert, so handelt es sich um die tiefere Frequenz und damit um log. 0; andernfalls liegt die höhere Frequenz (log. 1) an.

Das einfache Unterprogramm *(Bild 2)* für den Mikroprozessor 6502 geht davon aus, daß sich außer dem Nf-Eingang kein I/O-Port ändert. Die Adressenbelegung entspricht dem Mikrocomputer KIM-1, wobei der TTL-Nf-Eingang ein beliebiger Pin des Ports PB sein kann. Bei der Rückkehr aus dem Unterprogramm ist bei der höheren Frequenz das Zero-Flag gesetzt, bei der tieferen rückgesetzt, was leicht ausgewertet werden kann.

Die Abfrage nach 3/4 der Periodendauer der höheren Frequenz ist nur dann optimal, wenn sich die beiden FSK-Frequenzen etwa 1:2 verhalten. Andernfalls ist der Abfragezeitpunkt geringfügig zu ändern.

Übrigens hat die Software-Methode auch einen kleinen Nachteil etwa gegenüber Phase-Locked-Loop-Demodulatoren: Der erforderliche Störabstand muß mindestens etwa 6 dB betragen, während eine PLL noch mit 0...3 dB auskommt. Bei Tonband-Aufzeichnungen spielt dies überhaupt keine Rolle, da schon bei einfachen Kassettenrecordern der Störabstand über 30 dB liegt; bei Funk- oder Fernsprech-Übertragungen können sich jedoch manchmal Probleme ergeben. Hier ist es sinnvoll, selektive Filter und Begrenzerverstärker einzusetzen, um die Störbandbreite auf das für die Übertragung wirklich notwendige Frequenzband zu verringern und Amplitudenschwankungen zu beseitigen. Und während es wegen der unvermeidlichen Gleichlaufschwankungen bei Kassettengeräten meist besser ist, die Filterbandbreite nicht zu schmal zu machen, wird man bei Funkübertragungen die Bandbreite möglichst der Baudrate anpassen. Fe.

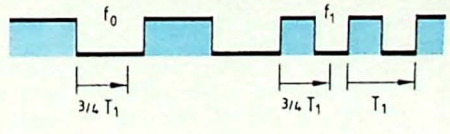


Bild 1. Fragt man ein Rechtecksignal zum richtigen Zeitpunkt ab, so läßt sich leicht feststellen, ob die höhere oder tiefere Frequenz des FSK-Signals anliegt

```
0200 AD 02 17 LDA 1702
0203 CD 02 17 CMP 1702
0206 FO FB    BEQ 0203
0208 A2 3B    LDX # 3B
020A CA       DEX
020B DO FD    BNE 020A
020D CD 02 17 CMP 1702
0210 60       RTS
```

Bild 2. Einfaches Unterprogramm zur FSK-Decodierung mit dem 6502. Liegt die höhere der beiden Frequenzen an, so ist bei der Rückkehr aus dem Unterprogramm im Status-Register das Zero-Flag gesetzt. Die Dimensionierung erfolgte für 1275/2125 Hz bei 1 MHz CPU-Taktfrequenz

Tondecoder sind Schaltungen, die das Vorhandensein eines Tones bestimmter Frequenz erkennen. Für diesen Zweck sind zwar preisgünstige PLL-ICs auf dem Markt, es ist aber dennoch reizvoll, eine solche Tonerkennung rein auf Software-Basis durchzuführen.

Ein Software-Tondecoder

Das Programm (*Bild 1*) wurde für den Mikrocomputer KIM-1 mit der CPU 6502 geschrieben, auf den sich auch die Adresse des I/O-Ports (PB 3) bezieht, an dem die zu prüfende Frequenz anliegt. Die Startadresse des Beispiel-Programms ist 0000.

Die Software-Tondecodierung besitzt einige deutliche Vorteile gegenüber herkömmlichen Hardware-Lösungen. So ist etwa die „Resonanzfrequenz" ebenso stabil wie der für die Mikrocomputer-Taktsteuerung verwendete Quarz; Frequenz und Bandbreite lassen sich in weiten Grenzen beliebig wählen, und der Zeitraum, der für das Decodieren eines Tones erforderlich ist, ist stets nur gerade so groß, wie es die programmierte Bandbreite erfordert.

```
TONDECODER-UNTERPROGRAMM
0040 A0 2D     LDY #2D
0042 A9 08     LDA #08
0044 88        DEY
0045 F0 06     BEQ 004D
0047 2C 02 17  BIT 1702
004A F0 F8     BEQ 0044
004C 88        DEY
004D F0 05     BEQ 0054
004F 2C 02 17  BIT 1702
0052 D0 F8     BNE 004C
0054 A0 05     LDY #05
0056 A2 16     LDX #16
0058 CA        DEX
0059 D0 FD     BNE 0058
005B 2C 02 17  BIT 1702
005E D0 0F     BNE 006F
0060 A2 2D     LDX #2D
0062 CA        DEX
0063 D0 FD     BNE 0062
0065 2C 02 17  BIT 1702
0068 F0 05     BEQ 006F
006A A2 2D     LDX #2D
006C 88        DEY
006D D0 E9     BNE 0058
006F 98        TYA
0070 60        RTS
BEISPIEL: EINSCHALTEN DES
KIM-DISPLAY BEI DER SOLL-
FREQUENZ
0000 20 40 00  JSR 0040
0003 D0 03     BNE 0008
0005 20 1F 1F  JSR 1F1F
0008 4C 00 00  JMP 0000
```

Bild 1. 6502-Maschinenprogramm zur Tondecodierung

Bild 2 macht die Arbeitsweise des Programms deutlich. Es wartet zunächst auf eine negative Flanke der Eingangsfrequenz; tritt eine solche innerhalb einer Periodendauer der Sollfrequenz nicht auf, so wird das Programm trotzdem fortgesetzt, um nicht „hängenzubleiben".

Dann wird nach einem Viertel der Soll-Periodendauer der Zustand des Eingangs abgefragt. Bei der richtigen Frequenz muß er noch Null sein, andernfalls erfolgt sofort ein Rücksprung.

Anschließend wird nun um eine halbe Soll-Periodendauer verzögert. Bei der richtigen Frequenz muß der Zustand des Eingangs dann 1 sein. Nach einer weiteren halben Periodendauer wird überprüft, ob der Eingang mittlerweile Null ist, und das Spiel wiederholt sich so oft, bis eine vorher programmierte Zahl von Perioden der Sollfrequenz durchlaufen ist. Entspricht der Zustand des Eingangs nun immer noch der „Vorhersage" des Mikrocomputers, so handelt es sich um die richtige Frequenz.

Die Bandbreite (B), innerhalb der ein Ton richtig erkannt wird, läßt sich mit der Zahl der überprüften Perioden (M) frei wählen. Bezeichnet man die Sollfrequenz mit f_o, so gilt die Beziehung

$$M = \frac{f_o}{2\,B}.$$

Da M (Adresse 0055) die hexadezimalen Werte 01...00 annehmen kann – 00 wird hier als dezimal 256 interpretiert –, ist die Güte (f_o/B) zwischen 2 und 512 frei wählbar. Bei einer Sollfrequenz von 1 kHz läßt sich daher eine Bandbreite von 2...500 Hz erzielen.

Die Frequenz wird mit den beiden hexadezimalen Variablen a (0057) und b (0061, 006B, 0041) festgelegt, die sich folgendermaßen errechnen:

$$a = \frac{1}{5}\left(\frac{10^6}{4\,f_o} - 7\right) \qquad b = \frac{1}{5}\left(\frac{10^6}{2\,f_o} - 9\right)$$

f_o kann dabei etwa zwischen 400 Hz und 30 kHz liegen (µP-Taktfrequenz: 1 MHz). Die Dimensionierung in der Programmauflistung erfolgte für eine Sollfrequenz von 2125 Hz und eine Bandbreite von rund 210 Hz. Die Decodierzeit ist von der Differenz zwischen der anliegenden und der Soll-Frequenz abhängig und beträgt maximal, d. h., wenn beide Frequenzen übereinstimmen, etwa

$$T = \frac{M + 2}{f_o}.$$

In unserem Beispiel ergibt sich ein Wert von etwa 3,3 ms. Es ist leicht einzusehen, daß die Decodierzeit der Bandbreite etwa umgekehrt proportional ist.

Bei der Rückkehr aus dem Unterprogramm sind Akku und Y-Register Null, wenn die richtige Frequenz erkannt wurde, und das Z-Flag im Status-Register ist 1. Das X-Register enthält dann den Wert b. Wurde nicht die Sollfrequenz decodiert, so ist das Z-Flag Null; der Akku und das Y-Register enthalten dann die Anzahl der Zyklen, die für ein richtiges Dekodieren noch gefehlt hatten. Damit ist auch eine Aussage dar-

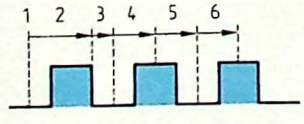


Bild 2. Prinzip der periodischen Abtastung. 1 = Programmstart, 2 = Warten bis zur nächsten negativen Flanke, 3 = Verzögerung um 0,25 Perioden, 4...6 = Verzögerung um je eine halbe Periode

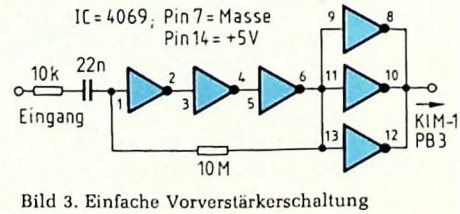


Bild 3. Einfache Vorverstärkerschaltung

über möglich, wie weit die Frequenz am Eingang vom Sollwert abweicht. Der Inhalt des X-Registers ist Null.

Das beschriebene Prinzip der Tondecodierung durch periodisches Abtasten wird auch als Sample-Methode bezeichnet. Zwei kleine, meist allerdings vernachlässigbare Nachteile des Prinzips sollen nicht verschwiegen werden: Der für ein richtiges Decodieren erforderliche Störabstand beträgt rund 6...10 dB, bezogen auf weißes Rauschen. Die Mikrocomputer-Tonerkennung ist damit guten PLL-Schaltungen um rund 6 dB unterlegen, was aber nur in Ausnahmefällen eine Rolle spielt. Den zweiten Nachteil hat die beschriebene Methode mit PLL-Schaltungen gemeinsam: Es gibt mehrere Frequenzen, die zu einem Ansprechen des Decoders führen können. So spricht das Sample-Prinzip auch auf 5 f_o, 9 f_o, 13 f_o usw. an, allerdings mit herabgesetzter Empfindlichkeit und Bandbreite. Im Gegensatz zur PLL werden jedoch keine „Subharmonischen" ausgewertet, Frequenzen also, deren Oberwellen zufällig die Sollfrequenz ergeben.

Selbstverständlich wird am Eingang des Mikrocomputers eine saubere Rechteckspannung mit möglichst symmetrischem Tastverhältnis benötigt. *Bild 3* zeigt eine geeignete Schaltung zur Aufbereitung von Eingangssignalen ab einigen mV.

Herwig Feichtinger

Morsedecoder in Basic

Die Programmiersprache Basic eignet sich normalerweise nicht allzusehr dazu, zeitkritische Aufgaben zu übernehmen. Entgegen der Annahme vieler Programmierer ist es aber möglich, Morsezeichen in Basic zu decodieren, wenn auch die erreichbare Geschwindigkeit vom verwendeten Computer bzw. dem Basic-Interpreter abhängt. Das im *Bild* wiedergegebene Programm entstand auf dem Mikrocomputer MZ-80 von Sharp. Zur Demonstration wird dabei die Shift-Taste als Morsetaste verwendet. Ist sie gedrückt, so ändert sich der Inhalt der Speicherzelle 57344 von 248 auf 249. Bei anderen Computern kann man z. B. einen Eingangsport abfragen, an den ein PLL-Tondecoder angeschlossen ist, um Telegrafie-Kurzwellensender empfangen zu können.

Wegen der hohen Anforderungen an die Arbeitsgeschwindigkeit des Programms dürfen keine FOR-NEXT-Schleifen verwendet werden. Rechnungen und Vergleiche erfolgen nur mit Variablen, nicht mit Zahlen, da sonst für die Umwandlung von Zahlen in das interne Fließkomma-Format zuviel Zeit verloren ginge.

Die Länge von Punkten, Strichen und Pausen wird in Zählschleifen ermittelt. Sinnvollerweise führt man alle notwendigen Berechnungen schon innerhalb dieser Schleifen aus, damit das Decodier-Ergebnis so schnell wie möglich vorliegt.

Die Decodierung wird nach folgendem Prinzip vorgenommen: In einem „Register" D(5) bis D(0) wird das Zeichen als Binärzahl aufgebaut. Je nach dem Ergebnis einer Zählschleife wird bei einem Punkt eine Null und bei einem Strich eine Eins eingeschoben sowie ein Zeiger um 1 erhöht. Durch entsprechenden Eintrag wird außerdem eine Null vor dem Zeichen hergeschoben – sonst wüßte man nicht, wo das Zeichen beginnt.

Sobald eine Pause genügend lang ist, wird diese Binärzahl als Zeiger benutzt, um das entsprechende Zeichen aus einer Liste herauszusuchen. Wenn die Pause andauert, wird zusätzlich ein Leerraum ausgegeben. Ob es sich um einen Punkt oder einen Strich handelt, wird durch Vergleich der Zählergebnisse mit einem Referenzwert ermittelt. Dieser Referenzwert muß sich an die Gebegeschwindigkeit anpassen. Dabei ist es nicht zweckmäßig, die letzte Punkt- oder Strichlänge als Bezug zu verwenden, weil der letzte Strich eines Zeichens häufig zu lang gegeben wird und das Tempo bei ungenauer Gebeweise „springt", so daß sich zu enge Toleranzen bei der Decodierung ergeben.

In diesem Programm läuft das Tempo der empfangenen Sendung nach, paßt sich also automatisch an. Das funktioniert natürlich nur dann, wenn nicht nur Punkte oder nur Striche gegeben werden. Zusätzlich könnte man auch noch die Pausendauer zur Anpassung heranziehen oder den Mittelwert mehrerer Zeichen verwenden.

Der Vergleich mit einem Programm, das in Maschinensprache geschrieben wurde, zeigte, daß bis zur maximal verarbeitbaren Geschwindigkeit von hier 100 BpM kaum Nachteile zu verzeichnen sind. Trotz alledem: Die Kombination Ohr-Gehirn-Erfahrung ist unschlagbar... *Volker Block*

```
1 REM VOLKER BLOCK DJ8JA
2 REM HANNOVERSCHE STR. 35
3 REM 3100 CELLE TEL. 05141/21342
5 REM EINGABE UEBER SHIFT-TASTE
6 REM WENN OBEN, ZELLE 57344=248 DEZ
7 REM TASTE PARALLEL ZUR SHIFT-TASTE
8 K0=0:K1=1:K2=2:K3=3:K4=4:K5=5:K6=6
9 T2=1.2:T8=1.8:K8=8:K7=16:K9=32
10 N1=248:P=57344:REM ENGABE-PORT
11 S4=7            :REM ANFANGS-TEMPO
80 DIM Z$(64)      :REM ZEICHENVORRAT
81 DIM D(5)        :REM AUFBAU-REGISTER
82 FOR I=0 TO 61:READ Z$(I):NEXT I
85 REM*********************************
100 D(K5)=K1:D(K4)=K1:D(K3)=K1:D(K2)=K1
101 REM                  WORTENDE
112 IF PEEK(P)=N1 THEN Z0=Z0+K1
113 IF Z0>K4*S4THEN PRINT" ";:GOTO 115
114 IF PEEK(P)=N1 GOTO 112
115 Z0=K5             :REM PAUSENZAEHLER
117 IF PEEK(P)=N1 GOTO 117 :REM WARTEN
118 Z2=K0             :REM REG.-POINTER
130 GOSUB 1000        :REM SIGNALDAUER
140 D(Z2)=K0: D(Z2-K1)=A: REM SHIFTEN
150 GOSUB 2000        :REM PAUSENDAUER
160 IF A=0 GOTO 130 :REM ELEMENTTRENN.
210 Z1=K0: PRINT Z$(S);:GOTO100
800 REM*********************************
900 REM            PUNKT-STRICH-LAENGE
1000 IF PEEK(P)=N1 GOTO 1010
1001 Z1=Z1+K1:GOTO1000
1010 A=K0             :REM ALSO PUNKT
1020 IF Z1>K2*S4 THEN A=K1:REM STRICH
1021 REM                 TEMPONACHLAUF
1022 IF Z1>K4*S4 THEN S4=S4*T2
1023 IF Z1<S4    THEN S4=S4/T2
1024 IF Z1>K6*S4 THEN S4=S4+S4
1025 IF Z1<S4/K2 THEN S4=S4/K2
1026 REM IF A=K1 THEN S4=Z1/K3
1030 Z1=K3:Z2=Z2+K1:IFZ2>K5 THEN Z2=K5
1900 RETURN
1940 REM*********************************
1950 REM                 PAUSENDAUER
2000 S=D(K5)*K9+D(K4)*K7+D(K3)*K8+D(K2)
 *K4+D(K1)*K2+D(K0):REM BINAER-DEZIMAL
2002 IF PEEK(P)=N1 GOTO 2010
2008 Z0=K5: A=K0:RETURN: REM ELEMENTTR.
2010 Z0=Z0+K1:IF Z0>T8*S4 THEN Z0=K5:
     A=K1:RETURN:REM BUCHSTABENENDE
2020 GOTO 2002
2030 REM*********************************
3000 DATA 5,6,"",7,"","","",8,"","",AR,
     "",?,"","",9,4,--,.,.,"",KA,"","
3001 DATA3,"","","",2,"",1
3010 DATA0,H,B,L,Z,F,C,P,"",V,X,"",Q,UE
   ,Y,J,"",S,D,R,G,U,K,W,O,I,N,A,M,E,T
3020 DATA"","","",""
4000 END
```

Basic-Programm zum Empfang von Morsezeichen. Zu Testzwecken wird hier die Shift-Taste des Computers MZ-80 als Eingang benutzt. Bei langsameren Basic-Interpretern sollte man alle Leerzeichen und REM-Kommentare weglassen

Die Bedienung eines Mikrocomputers über einen Fernschreiber ist aus verschiedenen Gründen wünschenswert. Bei der Eingabe von Programmen erhält die Bedienungsperson ein schriftliches Protokoll, das bei der Beseitigung von Fehlern im Programm eine große Hilfe darstellt. Die größere, schreibmaschinenähnliche Tastatur eines Fernschreibers ist bequemer und ermüdungsfreier zu bedienen als die oftmals gedrängt zusammenliegenden Tasten an Mikrocomputern.

Jörg Lohse

KIM-Monitorprogramm für Baudot-Fernschreiber

Bei der Auswahl von Fernschreibern gibt es grundsätzlich folgende zwei Möglichkeiten: Fernschreiber, die den ASCII-Code benutzen, und solche, die sich des Baudot-Codes bedienen. Die ASCII-Fernschreiber verfügen über Groß- und Kleinbuchstaben sowie Sonderzeichen. Der KIM-1 ist, wie viele andere Mikrocomputer, in der Lage, Zeichen im ASCII-Code zu senden und zu empfangen. ASCII-Fernschreiber haben allerdings den großen Nachteil, daß sie teuer sind.

Baudot-Fernschreiber beherrschen außer Ziffern und Zeichen nur Groß- oder Kleinbuchstaben. Der Code ist in Mikrocomputern nicht gebräuchlich; er ist komplizierter als der ASCII-Code, denn es müssen zwischen Buchstaben und Ziffern oder Zeichen Umschaltsignale gegeben werden. Aus dem öffentlichen Telexnetz stammende Baudot-Fernschreiber sind allerdings geprüft und instandgesetzt preiswert erhältlich. Sie kosten oft weniger als „Billigdrucker" für Mikrocomputer und verfügen im Gegensatz zu jenen über eine Tastatur und über direkten Netzanschluß.

Das folgende Programm erlaubt die Bedienung des KIM-1 mittels eines Baudot-Fernschreibers. Der Verfasser benutzte den Fernschreiber Siemens T 37, der häufig angeboten wird und die Benutzung von DIN-A-4-Papier erlaubt. Die Benutzung ähnlicher Fernschreiber (Lorenz LO-15, Siemens T-100 oder T-68) ist ebenfalls möglich.

Anschluß des Fernschreibers

Der Anschluß des Fernschreibers an den KIM-1 wird in *Bild 1* gezeigt. Das Programm benutzt zur Kommunikation die ASCII-Ein- und Ausgänge des KIM. T1 puffert den Ausgang des Mikrocomputers, da der Fernschreiber 40 mA Linienstrom benötigt. Die Dioden schützen die Logik auf der KIM-Platine und T1 vor Induktionsspitzen vom Empfangsmagneten. Da eine zum Empfangsmagneten antiparallel geschaltete Diode die Schaltzeiten verlängern würde, dienen die beiden hochspannungsfesten Dioden dem Schutz vor Induktionsspitzen.

Der Sendekontakt kann ohne besondere Maßnahmen mit dem Interface auf der KIM-Platine verbunden werden. Die Angabe der Farbe bei den Anschlußdrähten des Fernschreibers bezieht sich auf den Siemens T 37. Wenn ein anderer Typ zur Anwendung kommen soll, kann der Empfangsmagnet durch seinen Widerstand von 50...200 Ω ausfindig gemacht werden.

Das Fernschreiberinterface arbeitet auch, wenn das Monitorprogramm nicht läuft. Das Interface kann also vor der Eingabe des Programms geprüft werden: 5 V und 12 V werden an den KIM-1 geschaltet, und die Reset-Taste wird gedrückt. Dann kann der Fernschreiber mit der Netzspannung verbunden werden. Das Fernschreiberinterface erlaubt durch eine logische Verknüpfung auf der KIM-Platine, daß die auf der Tastatur bestätigten Buchstaben vom Drucker ausgegeben werden. Wenn mit der Tastatur auf dem Drucker geschrieben werden kann, haben Fernschreiber und Interface ihre Prüfung bestanden, und das Programm kann eingegeben und gestartet werden.

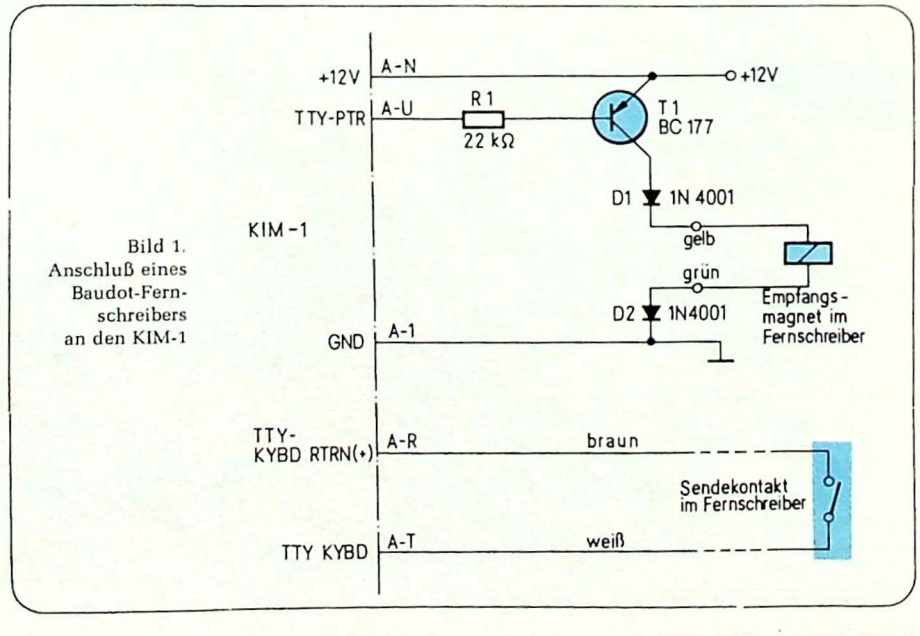


Bild 1. Anschluß eines Baudot-Fernschreibers an den KIM-1

Das Programm

Das Programm (*Bild 2*) ist nicht nur ein Monitor-Programm, welches in der Bedienung weitgehend identisch mit dem ASCII-Monitor im KIM-ROM ist. Es enthält auch für sämtliche Unterprogramme des KIM-ROMs, die mit dem ASCII-Fernschreiber zusammenarbeiten, jeweils ein entsprechendes Unterprogramm für Baudot-Fernschreiber, das in Anwenderprogrammen erscheinen kann. Innerhalb der Anwenderprogramme wird dabei ausschließlich mit ASCII-Zeichen gearbeitet, wie es bei Mikrocomputern üblich ist. In der *Tabelle* sind die Unterprogramme tabellarisch aufgeführt. Wenn ein Programm, welches für den Betrieb mit einem ASCII-Gerät geschrieben wurde, zusammen mit einem Baudot-Fernschreiber arbeiten soll, müssen die in der ersten Spalte der Tabelle im Programm vorkommenden Unterprogrammaufrufe durch jene in der 2. Spalte ersetzt werden.

```
0400 20 8C 1E 20 43 05 A9 4B 20 52 05 A9 49 20 52 05
0410 A9 4D 20 52 05 20 43 05 20 13 05 20 50 05 20 0C
0420 05 20 50 05 A9 00 85 F8 85 F9 20 8A 04 20 AC 1F
0430 C9 20 D0 06 20 CC 1F 4C 15 04 C9 4D D0 03 4C 4F
0440 1C C9 0D D0 06 20 63 1F 4C 15 04 C9 0A D0 0E 38
0450 A5 FA E9 01 85 FA B0 02 C6 FB 4C 15 04 C9 2E D0
0460 0C A0 00 A5 F8 91 FA 20 63 1F 4C 15 04 C9 47 D0
0470 03 4C C8 1D 4C 2A 04 20 8A 04 20 AC 1F 20 8A 04
0480 20 AC 1F A5 F8 60 EA EA EA EA 86 FD A9 80 2C 40
0490 17 D0 FB A2 00 86 FE 20 AE 05 A0 05 AD 40 17 29
04A0 80 46 FE 05 FE 85 FE 20 B4 05 88 D0 EF 46 FE 46
04B0 FE 46 FE 20 B4 05 A5 FE C9 1B D0 07 A9 20 85 EA
04C0 4C 8A 04 C9 1F D0 07 A9 00 85 EA 4C 8A 04 C9 02
04D0 D0 05 A9 0A 4C 07 05 C9 08 D0 05 A9 0D 4C 07 05
04E0 C9 04 D0 05 A9 20 4C 07 05 EA 05 EA 85 FE A2 FF
04F0 E8 BD C0 05 C5 FE D0 F8 86 FE A9 20 24 FE D0 06
0500 8A 09 40 4C 07 05 8A A6 FD A0 FF 60 A0 00 B1 FA
0510 4C 24 05 A5 FB 20 24 05 20 91 1F A5 FA 20 24 05
0520 20 91 1F 60 85 FC 4A 4A 4A 4A 20 35 05 A5 FC 20
0530 35 05 A5 FC 60 29 0F C9 0A 18 30 02 69 07 69 30
0540 4C 52 05 A9 0A 20 52 05 A9 0D 20 52 05 A2 FF 60
0550 A9 20 86 EC 20 8C 1E EA EA C9 0D D0 04 A9 08 D0
0560 0E C9 0A D0 04 A9 02 D0 06 29 3F AA BD C0 05 85
0570 EB 29 20 C5 EA F0 0E 85 EA A8 F0 04 A9 1B D0 02
0580 A9 1F 20 8D 05 A5 EB 20 8D 05 A6 EC 60 A2 06 8E
0590 42 17 20 B4 05 A0 04 4A 90 04 A2 07 D0 02 A2 06
05A0 8E 42 17 20 B4 05 88 10 EE A2 07 8E 42 17 A2 1D
05B0 20 B6 05 60 A2 13 8E 07 17 2C 07 17 10 FB 60 00
05C0 29 03 19 0E 09 01 0D 1A 14 06 0B 0F 12 1C 0C 18
05D0 16 17 0A 05 10 07 1E 13 1D 15 11 2D 00 3A 00 00
05E0 24 34 2C 29 2B 24 31 25 2F 32 39 31 2C 23 3C 3D
05F0 36 37 33 21 2A 30 35 27 26 38 2E 2B 24 3E 24 39
```

Bild 2. Hexadezimale Programmliste des Baudot-Monitors (05B5=Baudrate bzw. Bitdauer in ms)

Funktion des Monitorprogramms

Das Monitorprogramm ist in Page 4 und 5 untergebracht. Es wird an der Adresse 0400 gestartet und reagiert mit dem Ausdruck
KIM 0400 20.

Wenn die Adresse geändert werden soll, wird die neue Adresse eingetastet und dann die Leertaste gedrückt. Daraufhin werden vom Fernschreiber die neue Adresse und die darin stehenden Daten ausgedruckt. Sollen Daten geändert werden, so sind die neuen Daten und anschließend ein Punkt einzugeben. Das Monitorprogramm schreibt dann die neuen Daten in den Speicher und druckt die nächsthöhere Adresse aus. Wenn die nächsthöhere bzw. nächstniedrigere Adresse ausgedruckt werden soll, ist Wagenrücklauf bzw. Zeilenwechsel auszulösen. Wird „M" gedrückt, so folgt ein Sprung zum Systemprogramm des KIM-1. Wenn ein Programm an der zuletzt ausgedruckten Adresse gestartet werden soll, ist „G" zu betätigen. Bei der Eingabe von Adressen und Daten können Vornullen weggelassen werden.

Der Baudot-Monitor ist in der Bedienung mit dem im KIM vorhandenen ASCII-Fernschreib-Monitor fast identisch. Lochstreifenbefehle wurden in der Baudot-Version nicht vorgesehen, da Fernschreiber mit Lochstreifenzusatz ein Mehrfaches von dem kosten, was man für Modelle ohne Lochstreifenzusatz bezahlen muß, so daß die Speicherung von Programmen auf Kassettenrecordern preiswerter ist.

Das Programm arbeitet mit 50 Baud, so daß alle dem öffentlichen Telexnetz entstammende Fernschreiber benutzt werden können. Es kann auch in andere Bereiche des Speichers geladen werden, wenn die 3. Bytes von 3-Byte-Befehlen, falls diese 04 oder 05 lauten, entsprechend geändert werden. Dies läßt sich leicht mit einem Disassembler feststellen (vgl. FUNKSCHAU 1978, Heft 21).

Die Problematik der Ansteuerungs-Hardware für Fernschreiber sei hier nicht näher dargestellt, sondern stattdessen auf einen Beitrag namens „Fernschreiber-Ansteuerung" in FUNKSCHAU 1979, Heft 26, Seite 1531, verwiesen. Dort wird dargelegt, wie man auch ohne lebensgefährliche Hochspannungs-Netzteile, nämlich mit nur 12 V Betriebsspannung, genügend steile Stromanstiegs-Flanken erzielen kann.

Das Monitorprogramm wurde absichtlich in einen Speicher-Erweiterungsbereich gelegt, da es selbst einen nicht unerheblichen Adressenraum beansprucht und nicht als Selbstzweck, sondern für die Entwicklung von Anwenderprogrammen dienen soll. Die Verlegung in den KIM-RAM-Bereich kann daher nur eine Notlösung darstellen, wenngleich das speichermäßig durchaus möglich wäre.

Unterprogramme im Baudot-Monitor

ASCII-Startadresse	*Baudot-Startadresse*	*Name*	*Aufgabe*
1E2F	0543	CRLF	Wagenrücklauf, Zeilenwechsel
1E3B	0524	PRTBYT	Druckt Akkumulatorinhalt als 2 Hexadezimalzeichen aus
1E1E	0513	PRTPNT	Druckt Pointregister aus
1E9E	0550	OUTSP	Druckt einen Leerraum
–	050C	–	Druckt Daten unter der durch das Point-Register spezifizierten Adresse aus
1E5A	048A	GETCH	Holt ASCII-Zeichen vom Fernschreiber
1EA0	0552	OUTCH	Druckt Akkumulatorinhalt als ASCII-Zeichen
1F9D	0477	GETBYT	Holt 2 Hexadezimalzahlen vom Fernschreiber

Unter einer Software-Uhr versteht man ein Programm, das durch geeignete Verzögerungsschleifen einen „Software-Zähler" so weiterstellt, daß bestimmte Speicherzellen die aktuelle Uhrzeit enthalten, die dann z. B. auf einer Siebensegment-Anzeige ausgegeben werden kann.

Michael Klein

Software-Uhr für den Z-80-KIT

Das eigentliche Problem einer Software-Uhr liegt darin, daß sie in einer Endlosschleife läuft, in der es Verzweigungen gibt, aber dennoch nach einer Sekunde den Inhalt des Sekundenregisters erhöhen muß. Da durch die Sprünge das Programm einmal mehr und einmal weniger Schritte ausführen muß, gilt es die dadurch entstehende Zeitdifferenz durch geschickt angelegte Warteschleifen zu kompensieren. Aber auch hier gibt es Probleme, denn z. B. ein Sprungbefehl JR NZ e wird in 12 Taktzyklen ausgeführt, wenn die Sprungbedingung erfüllt ist; er braucht aber nur 7 Taktzyklen, wenn die Sprungbedingung nicht erfüllt ist. Das *Bild* zeigt ein Z-80-Programm. Allerdings wurde das Programm nicht so ausgelegt, daß die Schleife rechnerisch genau eine Sekunde für einen Durchlauf braucht. Es mußte auch die Zeitverzögerung mit einbezogen werden, die das WAIT-Flipflop verursacht. Dieses Wait-Flipflop ist auch für die Fehler verantwortlich, die bei der Zeitangabe auftreten. Deshalb wurde das Programm auch nicht auf 1 Mikrosekunde genau ausgetüftelt, obwohl dies möglich wäre. Das Programm enthält „Feh-

```
Sekundenspeicher:       3CDØ
Minutenspeicher:        3CD1
Stundenspeicher:        3CD2
Stackpointer:           3CFF
Programmzeiger:         3C5Ø

ANF: 3C5Ø   21   LD HL,3CDØ
       51   DØ
       52   3C
       53   ØE   LD C,6Ø
       54   6Ø
       55   CD   CALL LIMIT
       56   93
       57   3C
       58   2Ø   JR NZ DIS1
       59   14
       5A   36   LD (HL),ØØ
       5B   ØØ
       5C   23   INC HL
       5D   CD   CALL LIMIT
       5E   93
       5F   3C
       6Ø   2Ø   JR NZ DIS
       61   1Ø
       62   36   LD (HL),ØØ
       63   ØØ
       64   23   INC HL
       65   ØE   LD C,24
       66   24
       67   CD   CALL LIMIT
       68   93
       69   3C
       6A   2Ø   JR NZ DIS
       6B   Ø6
       6C   36   LD (HL),ØØ
       6D   ØØ
DIS1:3C6E   Ø6   LD B,Ø5
       6F   Ø5
A:     7Ø   1Ø   DJNZ A
       71   FE
DIS:   72   Ø6   LD B,6D
       73   6D
B:     74   1Ø   DJNZ B
       75   FE
       76   Ø6   LD B,Ø2
       77   Ø2
E:     78   C5   PUSH BC
       79   Ø6   LD B, B2
       7A   B2
       7B   C5   PUSH BC
D:     7C   21   LD HL,3CDØ
       7D   DØ
       7E   3C
       7F   FD   LD YI,ØØ11
       8Ø   21
       81   11
       82   ØØ
       83   Ø6   LD B,Ø3
       84   Ø3
C:     85   CD   CALL DISPL
       86   99
       87   3C
       88   1Ø   DJNZ C
       89   FB
       8A   C1   POP BC
       8B   1Ø   DJNZ D
8C   EE
8D   C1   POP BC
8E   1Ø   DJNZ E
8F   E8
9O   C3   JP ANF
91   5Ø
92   3C
;
; Unterprogramm LIMIT
;
3C93   7E   LD A,(HL)
  94   3C   INC A
  95   27   DAA
  96   77   LD (HL),A
  97   91   SUB C
  98   C9   RET
;
; Unterprogramm DISPL2
;
3C99   7E   LD A,(HL)
  9A   CD   CALL ROUT
  9B   A6
  9C   3C
  9D   7E   LD A,(HL)
  9E   CD   CALL SRL 4
  9F   19
  AØ   Ø1
  A1   CD   CALL ROUT
  A2   A6
  A3   3C
  A4   23   INC HL
  A5   C9   RET
;
; Unterprogramm ROUT
;
3CA6   E6   AND ØF
  A7   ØF
  A8   DD   LD IX, Ø3DE
  A9   21
  AA   DE
  AB   Ø3
  AC   16   LD D,ØØ
  AD   ØØ
3CAE   5F   LD E,A
  AF   DD   ADD IX,DE
  BØ   19
  B1   DD   LD A,(IX+d)
  B2   7E
  B3   ØØ
  B4   FD   LD C,(IY)
  B5   4E
  B6   ØØ
  B7   ED   OUT (C),A
  B8   79
  B9   FD   INC IY
  BA   23
  BB   ØØ   NOP
  BC   C9   RET
```

Dieses Programm macht aus dem Z-80-KIT eine Digitaluhr

ler“ von einigen hundert Mikrosekunden pro Durchlauf, die man jedoch mit einigem Zusatzaufwand loswerden könnte.

Das Programm ist so aufgebaut, daß es etwa 0,995 s lang die aktuelle Zeit anzeigt und während 0,005 s die neue Anzeige errechnet. Der Zeiger HL zeigt auf die Speicherzelle, deren Inhalt gerade behandelt wird. Das Einstellen von HL geschieht im ersten Programmschritt.

Dann wird der Grenzwert für die Sekundenanzeige gesetzt (60). Das Unterprogramm LIMIT erhöht den Inhalt der Speicherzelle, auf die HL zeigt und stellt fest, ob der Grenzwert 60 erreicht ist. Wenn nicht, verzweigt nach der Rückkehr aus dem Unterprogramm das Hauptprogramm nach der Abfrage JR NZ DIS1 nach rechts. Dann folgt eine Warteschleife. Sie bewirkt die gleiche Zeitverzögerung, wie wenn das Programm nach JR NZ DIS1 nach unten weiterverzweigt hätte. Das gleiche gilt auch für die beiden nachfolgenden Abfragen. Bei der letzten Abfrage, die nur jede Stunde erreicht wird, wurde die zusätzliche Warteschleife weggelassen. Die zweite Warteschleife mit dem Parameter B=6D dient zur Ergänzung des Programmes auf eine Sekunde „Zykluszeit“. Nun folgt der Teil des Programmes, in dem die Anzeige erfolgt. Hier sind zwei Zählschleifen eingebaut. Die ganz innen liegende Schleife mit B=3 dient lediglich dazu, alle drei Zeitwerte zur Anzeige zu bringen. Die beiden äußeren Schleifen mit B=B2 und B=02 braucht man, um die Anzeigedauer von ca. 0,995 s zu erreichen.

DISPL2 lädt den anzuzeigenden Wert in A und übergibt ihn an das Unterprogramm ROUT, das die eigentliche Anzeigensteuerung übernimmt. DISPL2 trennt das anzuzeigende Byte in das obere und untere „Nibble“ auf und übergibt es an ROUT. Je nachdem, welche Zeitkonstante der WAIT-Monoflop ausweist, kann eine Änderung der Konstanten 6D erforderlich werden. ■

HP-25 berechnet Frequenzgang

Programmierbare Taschenrechner besitzen meist nur eine eingeschränkte Zahl von Datenspeichern; beim HP-25 sind es acht. Trotzdem lassen sich damit mehr als acht Zahlenwerte speichern, wie das folgende Programmierbeispiel zeigt.

Ziel des Programms ist es, den Frequenzgang einer Schaltung zu errechnen, die maximal drei Tief- und/oder Hochpässe enthält, die voneinander entkoppelt sind. *Bild 1* macht die mathematischen Voraussetzungen für jeden Hoch- und Tiefpaß deutlich. Genaue Ergebnisse sind auch zu erwarten, wenn die drei Pässe nicht voneinander entkoppelt sind, sondern lediglich aufgrund ihrer Dimensionierung die Bedingungen von Bild 1 erfüllen, d.h. wenn der dem Generator nächste Pass niederohmig ausgelegt ist und die folgenden z.B. um den Faktor 10 hochohmiger werden.

Natürlich ist es nur mit einem Trick möglich, in den wenigen Datenspeichern des HP-25 alle benötigten Werte zu speichern. Dieser Trick besteht darin, die INT- und FRAC-Funktionen zum Zerlegen eines Speichers zu verwenden. Der Teil vor dem Dezimalpunkt speichert den Wert für R, der Nachkommateil denjenigen für C. Wie aus *Bild 2* hervorgeht, enthält das Vorzeichen der sich ergebenden Gesamtzahl außerdem die Aussage, ob es sich um einen Hoch- oder Tiefpaß handelt.

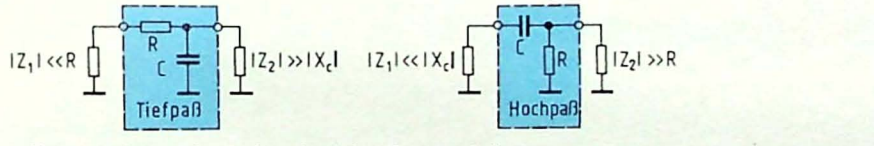
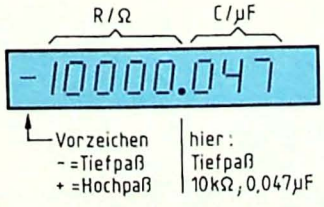


Bild 1. Die drei in der zu berechnenden Gesamtschaltung vorkommenden Elementar-Filter können ein R-C-Hochpaß oder ein R-C-Tiefpaß sein

Bild 3. ►
Das Programm belegt alle 49 Schritte des HP-25. Will man die Dämpfung von weniger als drei Elementarfiltern errechnen, so kann man in die nicht benutzten Datenspeicher (z. B. R 3) den Wert −1,000001 schreiben

01	STO 0	26	RCL 0
02	0	27	x
03	STO 6	28	g Pi
04	STO 7	29	x
05	RCL 6	30	2
06	g x=0	31	x
07	GTO17	32	g1/x
08	1	33	EEX
09	f x=y	34	6
10	GTO19	35	x
11	CLX	36	STO 4
12	2	37	g x≥0
13	f x=y	38	x⟷y
14	GTO21	39	÷
15	RCL 7	40	1
16	GTO00	41	STO+6
17	RCL 1	42	g →P
18	GTO22	43	f log
19	RCL 2	44	2
20	GTO22	45	0
21	RCL 3	46	x
22	ENTER	47	STO+7
23	f INT	48	PAUSE
24	x⟷y	49	GTO05
25	gFRAC		



◄ Bild 2.
In den HP-25-Registern R 1...3 werden sechs Bauelementewerte gespeichert: R 1 enthält die Werte des ersten Elementarfilters, R 2 die des zweiten und R 3 die des dritten

Bild 3 zeigt das Programm; es belegt alle 49 Programmschritte des HP-25. Nachdem es geladen ist, gibt man die Werte der drei Hoch- oder Tiefpässe in die Register 1...3 ein. Dann tippt man die zu untersuchende Frequenz in Hz ein und drückt R/S. Nacheinander erscheinen kurz die Einzeldämpfungen im Display, bis das Programm anhält und die Gesamtdämpfung bei der jeweiligen Frequenz in dB anzeigt. Nun kann man eine neue Frequenz eingeben, wieder R/S drücken usw.

Für alle, die gern anderer Leute Programme zerpflücken, hier noch die Belegung der übrigen Datenspeicher: In R 0 wird die eingegebene Frequenz, in R 4 die jeweilige Grenzfrequenz (neg. bei Tiefpaß) gespeichert; R 5 ist nicht belegt, R 6 gibt an, welcher Hoch- oder Tiefpaß gerade berechnet wird (0...2), und in R 7 werden die Dämpfungswerte in dB aufsummiert. *Fe.*

Oft findet der Computeranwender, der vielleicht einen AIM-65, einen KIM-1 oder einen SYM-1, also ein 6502-System besitzt, ein Programm, das ihn zwar brennend interessiert, das er aber auf seinem Computer nicht laufen lassen kann, weil es für einen anderen Prozessor geschrieben ist, etwa den 8080. Solange dieses Programm keine Echtzeitverarbeitung betreibt, kann dem mit einem Simulator, wie er im folgenden geschildert wird, abgeholfen werden.

Michael Konz

6502 simuliert 8080

Ein Simulator ist praktisch ein Interpreter; nur wird dabei keine höhere Programmiersprache wie BASIC in einen Maschinencode übersetzt, sondern ein Maschinencode in einen anderen [1, 2].

Daß die Ausführungszeiten des simulierten Prozessors dabei gewaltig steigen, ist selbstverständlich, kann aber oft hingenommen werden.

Dieser Simulator (*Bild 1*) bildet ein 8080-System softwaremäßig auf einem 6502-Mikrocomputer nach. Die Arbeitsregister des Pseudo-8080 befinden sich dabei in der Zero-Page, und zwar an folgenden Adressen:

Register	*Adresse*
Programmzähler PC	AA,AB
Stackpointer SP	A6, A7
Statusregister F	A8
Akku A	A9
H, L	A5, A4
D, E	A3, A2
B, C	A1, A0

```
0000 E9 0A 1A 02 025C 00 B1 02 BA 031C 66 A9 4C 10 03DC 91 A6 60 B1 049C 29 C7 85 A8 055C 20 01 06 68
0004 12 E3 F9 EB 0260 02 BF 02 C4 0320 03 A5 A8 4A 03E0 A6 48 A2 06 04A0 A5 A9 A2 00 0560 4C 40 05 20
0008 27 2F 37 3F 0264 02 C9 02 CF 0324 4C 0E 03 A5 03E4 20 38 03 B1 04A4 4A 90 01 E8 0564 68 03 A9 06
000C 07 0F 17 1F 0268 02 DE 02 E7 0328 A8 4C 1B 03 03E8 A6 48 20 38 04A8 C8 C0 08 90 0568 85 F1 6C F0
0010 00 C9 76 D3 026C 02 A0 05 F6 032C B1 AA 29 30 03EC 03 20 BD 03 04AC F7 8A 4A B0 056C 00 EA EA EA
0014 DB C3 CD 2A 0270 02 FD 02 04 0330 4A 4A 4A AA 03F0 68 95 A1 68 04B0 06 A5 A8 09 0570 B1 AA 29 38
0018 3A 22 32 C0 0274 03 0B 03 19 0334 60 20 2C 03 03F4 95 A0 60 20 04B4 06 D0 04 A5 0574 4A 4A 4A EA
001C C0 CF CF C7 0278 03 21 03 27 0338 F6 A0 D0 02 03F8 2C 03 B5 A0 04B8 A8 29 D3 85 0578 4A EA AA B5
0020 C7 CF CF CF 027C 03 0A 03 4E 033C F6 A1 60 20 03FC 65 A4 85 A4 04BC A8 A5 A9 08 057C 5D 24 A8 F0
0024 C7 C7 C7 C7 0280 03 62 03 63 0340 2C 03 38 B5 0400 B5 A1 65 A5 04C0 68 29 82 85 0580 01 60 90 02
0028 C7 CF 40 80 0284 05 63 05 76 0344 A0 E9 01 95 0404 85 A5 4C 10 04C4 AD 0A 0A 0A 0584 18 60 38 60
002C C1 C5 04 05 0288 03 80 03 93 0348 A0 B0 02 D6 0408 03 20 68 03 04C8 0A 0A 05 AD 0588 20 70 05 90
0030 03 0B 09 C0 028C 03 A0 03 A8 034C A1 60 B1 A6 040C 20 2C 03 A5 04CC 85 AD A5 A8 058C FA 4C 76 03
0034 06 C6 C2 C4 0290 03 B5 03 5D 0350 85 AA A2 06 0410 F0 95 A0 A5 04D0 29 17 05 AD 0590 20 70 05 90
0038 01 13 15 1B 0294 04 53 04 DF 0354 20 38 03 B1 0414 F1 95 A1 60 04D4 85 A8 60 A5 0594 F2 4C 80 03
003C 25 27 2A 01 0298 03 C7 03 2F 0358 A6 85 AB A9 0418 B1 AA 29 38 04D8 A8 4A 4C 77 0598 20 70 05 90
0040 02 03 01 02 029C 05 4F 05 35 035C 03 85 AF 4C 041C 4A 4A 4A C9 04DC 04 38 4C F7 059C EA 4C 4E 03
0044 03 76 04 D7 02A0 03 3F 03 F7 0360 38 03 A2 FF 0420 06 F0 09 AA 04E0 05 48 E5 AD 05A0 A5 A8 29 10
0048 04 DD 04 FE 02A4 03 98 05 6A 0364 9A 4C 82 E1 0424 B5 55 85 F0 04E4 85 A9 20 10 05A4 F0 22 24 AE
004C 04 06 05 12 02A8 04 3C 04 88 0368 C8 B1 AA AA 0428 84 F1 EA 60 04E8 03 20 04 03 05A8 10 17 A5 A9
0050 05 1E 05 25 02AC 05 90 05 09 036C C8 B1 AA 86 042C A5 A5 85 F1 04EC 68 28 29 0F 05AC 38 E9 06 85
0054 05 A1 A0 A3 02B0 04 A5 A4 85 0370 F0 85 F1 A0 0430 A5 A4 85 F0 04F0 48 A5 AD 29 05B0 A9 A5 A8 4A
0058 A2 A5 A4 00 02B4 AA A5 A5 4C 0374 00 60 20 68 0434 60 B1 AA 29 04F4 0F 85 AD 68 05B4 90 06 A5 A9
005C A9 40 01 04 02B8 7B 03 B1 A0 0378 03 86 AA 85 0438 07 4C 1F 04 04F8 E5 AD B0 9E 05B8 E9 60 85 A9
0060 80 52 47 55 02BC 85 A9 60 B1 037C AB 84 AF 60 043C 20 68 03 86 04FC 90 96 A5 A8 05BC E6 AE 4C A0
0200 A0 00 B1 AA 02C0 A2 85 A9 60 0380 A2 06 20 42 0440 AD B1 AA 29 0500 4A 90 DA 18 05C0 04 18 A5 A9
0204 A2 00 D5 00 02C4 A5 A9 91 A0 0384 03 A5 AB 91 0444 38 4A 4A AA 0504 90 D8 A5 A9 05C4 69 06 85 A9
0208 F0 25 E8 E0 02C8 60 A5 A9 91 0388 A6 20 42 03 0448 B5 45 85 F0 0508 25 AD 85 A9 05C8 A5 A9 29 0F
020C 1B 90 F7 B1 02CC A2 60 EA 20 038C A5 AA 91 A6 044C B5 46 85 F1 050C 20 12 03 4C 05CC C9 0A B0 F1
0210 AA 35 00 D5 02D0 D2 02 B1 A6 0390 4C 76 03 20 0450 6C F0 00 20 0510 A0 04 A5 A9 05D0 A5 A8 4A 90
0214 0F F0 18 E8 02D4 AA B9 A4 00 0394 68 03 B1 F0 0454 35 04 B1 F0 0514 45 AD 85 A9 05D4 06 A5 A9 20
0218 E0 2A 90 F3 02D8 91 A6 96 A4 0398 85 A4 C8 B1 0458 85 AD 4C 41 0518 20 12 03 4C 05D8 E3 05 EA A5
021C 20 F0 E9 A9 02DC C8 60 A5 A4 039C F0 85 A5 60 045C 04 20 35 04 051C 9A 04 A5 A9 05DC A9 C9 A0 B0
0220 45 20 7A E9 02E0 85 A6 A5 A5 03A0 20 68 03 B1 0460 B1 F0 48 20 0520 05 AD 4C 16 05E0 F4 90 DB 69
0224 A5 AB 20 46 02E4 85 A7 60 20 03A4 F0 85 A9 60 0464 18 04 68 91 0524 05 95 A9 48 05E4 5F 85 A9 4C
0228 EA A5 AA 4C 02E8 EA 02 B9 A4 03A8 20 68 03 20 0468 F0 60 20 68 0528 20 DD 04 68 05E8 FD 02 EA EA
022C 10 06 EA 8A 02EC 00 B6 A2 96 03AC AE 03 B9 A4 046C 03 8A 48 20 052C 85 A9 60 20 05EC EA 08 A9 00
0230 A2 00 D5 39 02F0 A4 99 A2 00 03B0 00 91 F0 C8 0470 18 04 68 91 0530 18 04 B1 F0 05F0 85 AE A5 A9
0234 90 03 E8 10 02F4 C8 60 A5 A9 03B4 60 20 68 03 0474 F0 60 18 4C 0534 48 29 0F C9 05F4 4C 7A 04 08
0238 F9 48 B5 3F 02F8 49 FF 85 A9 03B8 A5 A9 91 F0 0478 ED 05 48 65 0538 0F 20 01 06 05F8 A9 FF 85 AE
023C 85 AF 68 0A 02FC 60 A5 A8 09 03BC 60 20 2C 03 047C AD 85 A9 20 053C 68 18 69 01 05FC A5 A9 4C E1
0240 AA BD 5D 02 0300 03 85 A8 60 03C0 E0 06 90 02 0480 10 03 68 28 0540 91 F0 AA A5 0600 04 D0 06 A5
0244 85 F0 BD 5E 0304 A5 A8 49 01 03C4 E8 E8 60 20 0484 29 0F 48 A5 0544 A9 48 86 A9 0604 A8 09 12 D0
0248 02 85 F1 20 0308 85 A8 60 A5 03C8 BD 03 B5 A0 0488 AD 29 0F 85 0548 20 A0 04 68 0608 04 A5 A8 29
024C 5A 02 A5 AA 030C A9 0A 26 A9 03CC 48 B5 A1 48 048C AD 68 65 AD 054C 85 A9 60 20 060C C7 85 A8 60
0250 18 65 AF 85 0310 B0 EB A5 A8 03D0 A2 06 20 42 0490 29 10 F0 06 0550 18 04 B1 F0 0610 20 46 EA 4C
0254 AA 90 02 E6 0314 29 D6 85 A8 03D4 03 68 91 A6 0494 A5 A8 09 12 0554 38 E9 01 48 0614 62 03 E9 44
0258 AB 60 6C F0 0318 60 A5 A9 4A 03D8 20 42 03 68 0498 D0 04 A5 A8 0558 29 0F C9 0F
```

Bild 1. Hex-Dump des Simulatorprogramms. An den Adressen 0000...0060 sowie 025D...02B0 befinden sich Codetabellen. Die Simulatorroutine (Start 0200) ist als Unterprogramm ausgeführt

```
0F00 A9 LDA #00
0F02 85 STA AA
0F04 A9 LDA #08
0F06 85 STA AB
0F08 20 JSR 0200
0F0B 4C JMP 0F08
```

Bild 2. Einfaches 6502-Programm zum Ausführen von 8080-Maschinenprogrammen, die an der Adresse 0800 beginnen. Es ruft das Unterprogramm in Bild 1 zyklisch auf

Bei Stackoperationen, z. B. Unterprogramm-Aufrufen, darf man keinesfalls vergessen, den Stackpointer vorher zu laden, z. B. die Zelle A6 mit dem Wert D0 und A7 mit 01, so daß der Pseudo-8080 wie der 6502 Page 1 als Stack verwendet.

Außerdem werden die Plätze 00...60 der Zero-Page sowie AD, AE, AF, F0 und F1 benötigt. Das Programm selbst belegt die Speicherplätze 0200...0615, also etwas über 1 KByte. Die verwendeten System-Monitoradressen nennt *Tabelle 1*. Die Ausführung der Befehle erfolgt genauso wie beim 8080, bis auf wenige Ausnahmen. So sind die Befehle RST 0...7, DI und EI nicht vorhanden, da sie der Interruptverarbeitung dienen, die bei einer softwaremäßigen Simulation nicht in Betracht kommt.

Andere Funktionen haben die Befehle OUT, IN und HLT erhalten. Bei OUT und IN wird das Argument als niederwertiges Adreßbyte interpretiert. Das höherwertige Byte dazu ist immer 06. Es erfolgt ein Sprung auf die angegebene Adresse, wo eine Ein- oder Ausgaberoutine in der 6502-Sprache steht. *Tabelle 2* gibt dafür ein Beispiel.

Bei HLT wird die Ausführung des Programms abgebrochen, und es erfolgt die Rückkehr zum System-Monitor.

Tabelle 1: Systemadressen

Adr.	*AIM-65*	*KIM-1*	
021C	20 F0 E9	20 2F 1E	CRLF
0221	20 7A E9	20 A0 1E	OUTCH
0226	20 46 EA	20 3B 1E	OUTBYT
0610	20 46 EA	20 3B 1E	OUTBYT
0365	4C 82 E1	4C 4F 1C	START

Tabelle 2: Routine für „OUT 16"

Adr.	*AIM-65*	*KIM-1*
0616	A5 A9	A5 A9
0618	4C 46 EA	4C 3B 1E

(Der Akku des 8080 wird auf dem AIM-Display bzw. dem KIM-Terminal in Form zweier Hex-Ziffern ausgegeben.)

Dasselbe geschieht, wenn der Simulator auf einen Code stößt, der nicht im Befehlssatz des 8080 enthalten ist, oder auf RST, DI und EI. In diesen Fällen wird zusätzlich die Adresse des fehlerhaften Bytes ausgegeben, d. h. der augenblickliche Stand des 8080-PC.

Der Simulator ist als Unterprogramm ausgeführt. Bei jedem Aufruf wird nur ein einziger Befehl ausgeführt. Daher ist es sinnvoll, einen kleinen Monitor zu schreiben, der nicht nur die Abarbeitung von Programmen, sondern auch das Setzen des Stackpointers und die Anzeige der 8080-Register übernimmt. Auf die Wiedergabe eines solchen Monitors wurde hier verzichtet, um nicht den universellen Charakter des Simulators für alle 6502-Systeme zu zerstören. In dieser Form läuft er nämlich ohne Änderungen auf allen Systemen, wenn sie nur über ausreichenden Speicherraum verfügen, da der System-Monitor nur bei der Fehlermeldung bemüht wird.

Soll das Programm in einen anderen Speicherbereich transferiert werden, so müssen auch die Tabellenwerte von 025D bis 02B0 und von 0045 bis 0054 entsprechend geändert werden; sie enthalten Sprungadressen.

Die Ausführungszeiten des Simulators liegen bei 117...1259 µs (Taktfrequenz: 1 MHz). Im Durchschnitt ist also etwa mit einer Verhundertfachung der 8080-Befehlszyklen zu rechnen. Sehr lange und zeitkritische 8080-Programme sollten daher lieber für den 6502 umgeschrieben werden, will man nicht eine extrem lange Laufzeit in Kauf nehmen. Zum Umschreiben aber ist das

```
0800 21 01 07  MVI HL,0701
0803 3A 00 07  LDA 0700
0806 86        ADD (HL)
0807 27        DAA
0808 32 02 07  STA 0702
0808 76        HLT
```

Bild 3. Dieses 8080-Programm addiert die Inhalte der Speicherzellen 0700 und 0701 und speichert das dezimale Ergebnis bei 0702 ab

Verständnis des Programmablaufs erforderlich, und hierbei kann wiederum der Simulator eine wertvolle Hilfe sein.

Um den Einsatz der Simulator-Routine zu verdeutlichen, zeigt *Bild 2* ein kleines Programm, das den 8080-Programmzähler auf 0800 setzt und den dort stehenden 8080-Maschinencode abarbeitet, bis er auf HLT oder einen anderen Stopbefehl trifft. *Bild 3* enthält – wiederum als Beispiel – ein kurzes 8080-Programm, das den Inhalt der Adresse 0700 zum Inhalt der Adresse 0701 addiert, in den BCD-Code umformatiert (DAA) und bei 0702 abspeichert. Das Programm läuft in dieser Form unverändert auf dem AIM-65 und auf dem KIM-1, wobei allerdings jeweils die 4-KByte-Version vorhanden sein muß.

Literatur

[1] 8080-Operationscodes. FUNKSCHAU 1979, Heft 26.

[2] Der Individualisten-Prozessor. Sonderheft Hobbycomputer 2, Franzis-Verlag, München 1979. ■

KIM spielt Lotto

Das im *Bild* als Hex-Dump abgedruckte 6502-Programm für den Mikrocomputer KIM-1 verwirklicht die „Ziehung" von Zufallszahlen nach dem Zahlenlotto-System, d.h. es darf keine Wertewiederholung auftreten. Die Startadresse ist 0200; die einzelnen Ziehungen werden mit der Taste 0 ausgelöst. Solange der Tastendruck anhält, durchläuft das Programm in schneller Folge und für den Benutzer unsichtbar die Zahlen 1...49. Beim Loslassen der Null-Taste prüft der Mikrocomputer, ob die nun ermittelte Zufallszahl schon einmal gezogen wurde; wenn ja, wird die Lottozahl dezimal um 1 erhöht, und es erfolgt noch einmal ein Test, ob die Zahl bereits gezogen wurde.

Auswahlwette-Spieler können das Programm auch verwenden; dazu muß nur die Adresse 020C mit 45 geladen werden. In der Zero-Page werden außer dem Display-Bereich (F9...FB) noch die Zellen 0...8 verwendet. Viel Glück beim Ziehen! *Raimund Pourvoyeur*

```
0200 F8 A9 00 85 06 A9 99 85 08 A5 08 C9 49 D0 02 A9
0210 00 18 69 01 85 08 20 6A 1F C9 00 F0 3D A5 07 C9
0220 00 D0 21 A5 08 C5 00 F0 E0 C5 01 F0 DC C5 02 F0
0230 D8 C5 03 F0 D4 C5 04 F0 D0 A5 06 8D 41 02 A5 08
0240 85 02 E6 06 A9 FF 85 07 A5 06 85 FA A9 00 85 FB
0250 A5 08 85 F9 20 1F 1F 4C 16 02 A9 00 85 07 85 FB
0260 85 FA 85 F9 20 1F 1F A5 06 C9 06 D0 09 A9 00 A2
0270 06 95 00 CA 10 FB 4C 09 02
```

Ob der KIM-1 beim Lottospielen mehr Glück hat als sein Operator, bleibt abzuwarten – jedenfalls erspart einem dieses Programm das Grübeln über die „richtigen" Zahlen

Fernschreiber sind meist ebenso laute wie unförmige papierverschwendende Geräte. Wenn man auf eine Verewigung empfangener Fernschreibsendungen verzichtet – was besonders im Amateurfunk oft vertretbar ist – so läßt sich das Siebensegment-Display des Mikrocomputers KIM-1 dazu mißbrauchen, RTTY mitzulesen.

KIM als Siebensegment-Fernschreiber

Ausgangspunkt des hier vorgestellten Programms ist das im HOBBY-COMPUTER-Sonderheft des Franzis-Verlages abgedruckte Siebensegment-Pseudoalphabet für den KIM-1. Es stellt den Inhalt von sechs aufeinanderfolgenden Speicherzellen als alphanumerischen Siebensegment-Code auf dem sechsstelligen KIM-Display dar.

```
0000 A9 00     LDA #00
0002 85 F3     STA F3
0004 A9 08     LDA #08
0006 85 F4     STA F4
0008 20 60 00  JSR 0060
000B A9 08     LDA #08
000D 2C 02 17  BIT 1702
0010 D0 F2     BNE 0004
0012 C6 F4     DEC F4
0014 10 F2     BPL 0008
0016 A9 05     LDA #05
0018 85 F5     STA F5
001A A9 13     LDA #13
001C 85 F4     STA F4
001E 20 60 00  JSR 0060
0021 C6 F4     DEC F4
0023 10 F9     BPL 001E
0025 18        CLC
0026 A9 08     LDA #08
0028 2C 02 17  BIT 1702
002B F0 01     BEQ 002E
002D 38        SEC
002E 66 F0     ROR F0
0030 C6 F5     DEC F5
0032 10 E6     BPL 001A
0034 A5 F0     LDA F0
0036 6A        ROR A
0037 6A        ROR A
0038 29 1F     AND #1F
003A C9 1F     CMP #1F
003C F0 C2     BEQ 0000
003E C9 1B     CMP #1B
0040 D0 04     BNE 0046
0042 A9 20     LDA #20
0044 D0 BC     BNE 0002
0046 05 F3     ORA F3
0048 AA        TAX
0049 BD A0 00  LDA 00A0,X
004C 48        PHA
004D A2 04     LDX #04
004F B5 E0     LDA E0,X
0051 95 E1     STA E1,X
0053 CA        DEX
0054 10 F9     BPL 004F
0056 68        PLA
0057 85 E0     STA E0
0059 4C 04 00  JMP 0004
```

Bild 1. Programmteil zum Empfang von Baudot-Fernschreibzeichen. Die Werte an den Adressen 0005 (halbe Startbitlänge) und 001B (Bitlänge) sind für die Baudrate maßgebend (hier 50 Bd)

Das Programm (*Bild 1*) zum Empfang der Fernschreibzeichen geht davon aus, daß an Port PB 3 ein Empfangskonverter angeschlossen ist, der dem Mark-Signal den H-Pegel (TTL) und dem Space-Signal den Low-Pegel zuordnet. Geeignete Schaltungen hierfür finden sich in FUNKSCHAU 1978, Heft 14, und in FUNKSCHAU 1979, Heft 9, wobei lediglich u. U. durch Zwischenschalten eines Inverters die richtige Polarität herzustellen ist.

Eine Besonderheit des Programms ist, daß die Zeichen nur dann ausgewertet werden, wenn das Startbit mindestens die halbe Datenbitlänge aufweist. Dadurch werden kurzzeitige Störimpulse wirkungsvoll unterdrückt.

Die schon erwähnte Display-Routine wurde hier lediglich als Hex-Liste abgedruckt (*Bild 2*). Sie wurde nur geringfügig geändert, so daß sie rund 1 ms dauert und dadurch auch als „Timer" zur Verzögerung definierter Zeiten verwendet werden kann. Der Adressenbereich 00A0...00DF enthält eine Umsetztabelle vom Baudot-Code in den Siebensegment-Code; die Reihenfolge der Zeichen entspricht darin der in ELEKTRONIK 1978, Heft 14, Seite 62 abgedruckten Tabelle, so daß Änderungen leicht möglich sind.

Sicher ist das Pseudoalphabet bei der ersten Inbetriebnahme des Fernschreibprogramms etwas ungewohnt. Der Text läuft mit der vollen Geschwindigkeit der empfangenen Zeichen von rechts nach links durch. Es besteht aber kein Grund, sich anfangs von dem „lustigen Flackern" einschüchtern zu lassen. Bei einiger Übung im Lesen des Pseudoalphabets – Cyrillisch kann man ja schließlich auch erlernen – ist das Mitlesen auch bei vollen 45 oder 50 Baud noch kein Problem. Bei Amateur-Funkfernschreiben (z. B. im KW-Bereich oder 2-m-Band) liegt die durchschnittliche Geschwindigkeit ohnehin meist niedriger.

Es sei noch erwähnt, daß auch hier im Prinzip eine Geschwindigkeitsumsetzung mittels eines Bufferspeichers möglich wäre, wie dies in FUNKSCHAU 1979, Heft 7, unter dem Titel „ASCII-Eingabe – ganz nebenbei" beschrieben wurde. In der Praxis hat sich dies beim Empfang von u. U. gestörten Fernschreibsendungen jedoch nicht bewährt, da man die Auswirkungen einer Falschabstimmung des Empfängers erst nach einigen Sekunden auf dem Display zu Gesicht bekommt, was die Abstimmung erheblich erschwert.

Gerhard Kirchberger
Herwig Feichtinger

```
0060 A9 7F 8D 41 17 A2 09 A0 06 B9 DF 00 84 FC A0 00
0070 8C 40 17 8E 42 17 8D 40 17 A0 18 88 D0 FD E8 E8
0080 A4 FC 88 D0 E4 8E 42 17 A9 00 8D 41 17 60 00 00

00A0 80 F9 80 F7 80 ED 85 BE 80 DE D0 9E D4 F1 D8 F8
00B0 B1 DB B8 FE F4 EE F3 E7 DC FC BD 80 B7 F6 9C 80
00C0 80 CF 80 80 80 80 FF 87 80 80 E6 80 80 80 C1 80
00D0 E6 80 80 DB 80 FD BF 86 EF D3 80 80 80 80 C1 80

00F0 ZEICHENBUFFER
00F3 BU/ZI-FLAG
00F4 ZEITGEBER
00F5 BITZAEHLER
00E0...00E5 DISPL.
```

Bild 2. Das Display-Unterprogramm (0060...008D) und die Baudot-Siebensegment-Umsetztabelle (00A0...00DF)

Von A bis Z

Alphabetisches Sortieren mit dem 6502

Eine Kartei ist eine Datensammlung in meist alphabetischer Anordnung. Eine solche Anordnung ist zwar bei einer Computerkartei [1] zum Finden eines Stichwortes nicht mehr von so ausschlaggebender Bedeutung wie bei einer auf Karten vorliegenden, sie kann aber nützlich sein, wenn man eine Kartei durchmustern oder ausdrucken lassen möchte [2]. Das folgende Programm ordnet eine Kartei, die z. B. mit Hilfe des Programms aus Heft 24/78 erstellt wurde, in alphabetischer Reihenfolge. Im Gegensatz zu ähnlichen Programmen in BASIC benötigt dieses Maschinenprogramm z. B. für einen 4-KByte-Block nur wenige Sekunden.

Arbeitsweise des Programms

Das Programm vergleicht, am Textanfang beginnend, aufeinanderfolgende Zeilen miteinander. Als Zeilen werden hier bis zu 256 Zeichen lange Strings verstanden, die mit CR-Zeichen abgeschlossen sind. Stößt es auf zwei Zeilen, die in der falschen Reihenfolge stehen, so tauscht es die beiden aus: Zeile 1 wird in einen Pufferspeicher geladen, Zeile 2 auf den Platz von Zeile 1 verschoben und Zeile 1 hinter die neue Zeile 1 gehängt. Als Abschluß der Austauschaktion wird ein Flag gesetzt. Nun wird die neue Zeile 2 mit der nächsten Zeile verglichen, gegebenenfalls ausgetauscht usw.

Das Verfahren wird wiederholt, bis der Rechner das Textende erkennt (hex 3 E). Hier wird das Austauschflag abgefragt. Wenn im ganzen Durchgang, vom Anfang bis zum Ende des Textes, keine Austauschoperation vorgenommen wurde, ist das Programm beendet – alle Zeilen stehen in der richtigen Reihenfolge und es erfolgt ein Sprung zum KIM-Monitor[1]). Andernfalls springt das Programm an seinen Anfang zurück (0000).

Zeilenvergleich

Es werden nur Buchstaben und Ziffern zum Vergleich der Zeilen benützt, alle anderen Zeichen werden überlesen. Dazu dient ein Unterprogramm an der Adresse 0098. So wird beim Vergleich der Zeilen

PREIS...DM 4.70
PREIS...DM 42.00

die zweite Zeile an den Anfang gestellt, da für den Rechner die Zeilen bis zur 4 identisch sind, danach die 2 aber kleiner ist als die 7. Sind zwei unterschiedlich lange Zeilen bis zum Ende der kürzeren identisch, so wird die kürzere jeweils vorangestellt; Vertauscht würden z. B.

FUNKSCHAU-SONDERHEFT
FUNKSCHAU

Zeilen mit Ziffern am Anfang werden nach den Ziffern sortiert und den mit Buchstaben beginnenden Zeilen vorangestellt. Wie in [1] geht das Programm davon aus, daß der Text keine Kleinbuchstaben enthält; will man dies ändern, so ist der Inhalt der Zelle 00A6 auf 7B abzuändern.

Speicherbedarf

Das Programm liegt im Speicherbereich 0000 bis 00AF. In Seite Null werden die Zellen E0...E8 als Arbeitsspeicher benötigt. Außerdem ist ein Pufferspeicher erforderlich, der hier bei 0100 beginnt und dessen Umfang die maximale Zeilenlänge bestimmt (maximal 256 Zeichen). Das Programm prüft nicht nach, ob es den Pufferspeicher überschreibt, hier ist also Vorsicht geboten.

Zusätzlich sind noch zwei Speicherstellen mit der Anfangsadresse des Textbereiches belegt. Im Programm sind dies die Zero-Page-Zellen B0 und B1. Sie werden nur bei Adresse 0002 abgefragt und können bei Bedarf leicht durch Modifikation von Speicherzelle 0003 auf einen anderen Platz der Seite Null gelegt werden. Eine Änderung der Sortierrichtung ist zu erreichen, indem man bei Adresse 004E den Befehl BCS (hex B0) durch BCC (hex 90) ersetzt.

Das Programm läuft ohne Änderung auf dem KIM-1. Auf dem AIM-65 läßt es sich vorteilhaft zusammen mit den Editor-Befehlen des Monitorprogramms verwenden. Es ist darauf zu achten, daß der erste Text nicht mit CR beginnt und der letzte Text mit CR (hex 0D) und „>" (hex 3E) abgeschlossen ist. Die Textanfangsadresse in den Zellen 00B0 und 00B1 zeigt hier auf 0202, was dem Datensuche-Programm [1] entspricht.

Hans-Herbert Hagedorn

[1]) Beim AIM-65 sollte man den Buffer bei 0170 beginnen lassen, wenn die Strings nicht länger als etwa 100 Zeichen sind.

Literatur

[1] KIM auf Datensuche. FUNKSCHAU 1978, Heft 24

[2] Textausdruck mit Baudot-Fernschreiber. FUNKSCHAU 1979, Heft 16


```
0000 A2 02     LDX #02
0002 B5 AF     LDA AF,X
0004 95 E4     STA E4,X
0006 95 E6     STA E6,X
0008 CA        DEX
0009 D0 F7     BNE 0002
000B A0 00     LDY #00
000D 84 E2     STY E2
000F 84 E3     STY E3
0011 B1 E7     LDA (E7),Y
0013 E6 E7     INC E7
0015 D0 02     BNE 0019
0017 E6 E8     INC E8
0019 C9 0D     CMP #0D
001B D0 F4     BNE 0011
001D A4 E2     LDY E2
001F B1 E5     LDA (E5),Y
0021 C9 0D     CMP #0D
0023 F0 68     BEQ 008D
0025 20 98 00  JSR 0098
0028 90 F5     BCC 001F
002A 84 E2     STY E2
002C 85 E1     STA E1
002E A4 E3     LDY E3
0030 B1 E7     LDA (E7),Y
0032 C9 3E     CMP #3E
0034 D0 09     BNE 003F
0036 C5 E4     CMP E4
0038 F0 73     BEQ 00AD
003A 85 E4     STA E4
003C 4C 00 00  JMP 0000
003F C9 0D     CMP #0D
0041 F0 0D     BEQ 0050
0043 20 98 00  JSR 0098
0046 90 E8     BCC 0030
0048 84 E3     STY E3
004A C5 E1     CMP E1
004C F0 CF     BEQ 001D
004E B0 3D     BCS 008D
0050 A0 00     LDY #00
0052 B1 E5     LDA (E5),Y
0054 99 00 01  STA 0100,Y
0057 C9 0D     CMP #0D
0059 F0 04     BEQ 005F
005B C8        INY
005C 4C 52 00  JMP 0052
005F A0 00     LDY #00
0061 B1 E7     LDA (E7),Y
0063 91 E5     STA (E5),Y
0065 C9 0D     CMP #0D
0067 F0 04     BEQ 006D
0069 C8        INY
006A 4C 61 00  JMP 0061
006D C8        INY
006E 98        TYA
006F 18        CLC
0070 65 E5     ADC E5
0072 85 E7     STA E7
0074 A5 E6     LDA E6
0076 69 00     ADC #00
0078 85 E8     STA E8
007A A0 00     LDY #00
007C B9 00 01  LDA 0100,Y
007F 91 E7     STA (E7),Y
0081 C9 0D     CMP #0D
0083 F0 04     BEQ 0089
0085 C8        INY
0086 4C 7C 00  JMP 007C
0089 A9 FF     LDA #FF
008B 85 E4     STA E4
008D A5 E8     LDA E8
008F 85 E6     STA E6
0091 A5 E7     LDA E7
0093 85 E5     STA E5
0095 4C 0B 00  JMP 000B
0098 C8        INY
0099 C9 30     CMP #30
009B 90 0C     BCC 00A9
009D C9 3A     CMP #3A
009F 90 0A     BCC 00AB
00A1 C9 41     CMP #41
00A3 90 04     BCC 00A9
00A5 C9 5B     CMP #5B
00A7 90 02     BCC 00AB
00A9 18        CLC
00AA 60        RTS
00AB 38        SEC
00AC 60        RTS

00AD 4C 4F 1C  JMP KIM

00B0 02 TEXT-ANFANG L
00B1 02 TEXT-ANFANG H
```

6502-Programm zum Sortieren von Textblöcken nach dem Alphabet

Für das Erlernen von Morsezeichen hat mancher Funkamateur schon kostbare Monate seines Lebens verwendet, und auch mit einiger Übung schafft man erst, etwa 60...100 Buchstaben pro Minute aufzunehmen. Mit Mikrocomputern lassen sich Morsezeichen dagegen bequem auch noch z. B. bei 300 BpM aufnehmen. Das vorliegende Programm stellt sie auf dem Display des KIM-1 als durchlaufende Schrift dar.

KIM decodiert Morsezeichen

Hardware-Voraussetzungen

Zum Empfang von Morsezeichen benötigt man selbstverständlich erst einmal einen Empfänger, z. B. für die Kurzwellen-Amateurfunkbereiche, an dessen Ausgang eine Schaltung angeschlossen wird, die die empfangenen Morsezeichen in ein verwertbares Digitalsignal umformt. Eine solche Schaltung befindet sich in FUNKSCHAU 1979, Heft 9, Seite 530, unter dem Titel „Ein einfacher Fernschreibkonverter". Sie verwendet eine PLL-Schaltung (NE 567) mit „Open-Collector"-Ausgang. Pin 8 des IC wird nun einfach mit dem KIM-Eingangsport PB 0 verbunden. Das ist alles, was man außer dem KIM selbst zum Empfang der Morsezeichen benötigt. Unter Umständen lassen sich mit aufwendigeren Filterschaltungen zwar bessere Empfangsergebnisse erzielen, doch genügt die Störsicherheit einer PLL-Schaltung in den meisten Fällen vollkommen. Als Kontrollanzeige sollte man die Serienschaltung einer LED und eines Widerstandes von 1 kΩ von Pin 8 des IC nach +5 V legen.

Nichts ist perfekt

Es wäre übertrieben, zu sagen, daß ein Mikrocomputer das Erlernen von Morsezeichen überflüssig macht. Wer das hier vorgestellte Programm ausprobiert, wird bald merken, daß Mikrocomputer sehr schnell ihre Grenzen finden; auch Signale, die man „mit dem Ohr" noch gut mithören kann, können bereits zu Schwierigkeiten führen, besonders, wenn die sendende Station eine recht unsaubere Gebeweise hat und dadurch das Zeitschema des Mikroprozessors durcheinanderbringt. Dagegen gibt es selten Schwierigkeiten, wenn auch die Gegenstation einen Mikrocomputer zum Senden verwendet (vgl. FUNKSCHAU 1979, Heft 14, KIM als Morseschreibmaschine) oder mit einer elektronischen, automatischen oder halbautomatischen Taste sendet.

Bezieht man das „Timing" der Morsezeichen auf die Dauer eines Punktes, so ist der Zwischenraum zwischen den Elementen eines Buchstabens ebenfalls einen Punkt lang; ein Strich besitzt drei Punktlängen, und der Zwischenraum zwischen zwei Buchstaben ist auch drei Punkte lang. Zwischen zwei Worten macht man üblicherweise mindestens fünf Punktlängen Pause.

So funktioniert das Decodieren

Das *Bild* zeigt das hexadezimal aufgelistete Programm für den KIM-1. Die Startadresse ist 0000. Wer statt PB 0 einen anderen Port als Eingang verwenden möchte, kann die entsprechende Maske für PB an den Adressen 000E, 0050 und 0066 ändern.

Zum Decodieren mißt das Programm zunächst die Länge des empfangenen Zeichenelementes (Adresse 0004). Als „Timer" wird dabei die Verzögerung einer alphanumerischen Anzeigeroutine (00A0) verwendet, die weitgehend der Routine aus dem Hobbycomputer-Sonderheft des Franzis-Verlages (S. 38) entspricht. Ist das Zeichen kürzer als zwei Punktlängen, so wird es als Punkt, andernfalls als Strich interpretiert. Wenn auf dem KIM-Tastenfeld die Taste 0 gedrückt wird, so schaltet das Programm auf automatische Geschwindigkeitsanpassung um: Wurde das Zeichenelement als Punkt erkannt, so wird die gerade gemessene Zeit als neue Punktlänge übernommen. Störimpulse, die kürzer als eine halbe Punktlänge sind, werden allerdings ignoriert (006C); ihre Länge wird zur Pausezeit addiert. Dies hat jedoch zur Folge, daß die Automatik nicht über den ganzen Variationsbereich arbeitet. Vor dem Einschalten der Automatik muß deshalb das Empfangstempo mit einer der manuellen Geschwindigkeitswahltasten auf den Fangbereich der Tempoautomatik vorgesetzt werden. Um diesen Nachteil zu vermeiden, müßte auf eine Störimpulsunterdrückung verzichtet

Hexadezimale Auflistung des Morse-Decodierprogramms für den KIM-1. Die Startadresse ist 0000

```
0000 A9 01 85 EE A9 FF 85 ED 20 A0 00 E6 ED A9 01 2C
0010 02 17 F0 F4 A5 ED C5 EC 90 EA C5 EB B0 1E 06 EE
0020 A6 E6 D0 1C A6 ED A4 ED CA 88 88 10 FB 86 EC 18
0030 A5 ED 65 ED 85 EB 65 EB 85 EA D0 04 06 EE E6 EE
0040 A9 FF 85 E8 20 A0 00 E6 E8 A5 EA C5 E8 90 34 A9
0050 01 2C 02 17 D0 EE A9 FF 85 E7 20 A0 00 E6 E7 A5
0060 E7 C5 EC B0 11 A9 01 2C 02 17 F0 EE A5 E7 18 65
0070 E8 85 E8 4C 44 00 A5 E7 85 ED A5 EB C5 E8 90 13
0080 4C 08 00 20 1D 01 20 27 01 20 1D 01 A9 00 85 E0
0090 4C 00 00 20 1D 01 20 27 01 A9 01 85 EE 4C 0D 00
00A0 A2 03 86 E9 A9 7F 8D 41 17 A2 09 A0 06 B9 DF 00
00B0 84 DF A0 00 8C 40 17 8E 42 17 8D 40 17 A0 20 88
00C0 D0 FD E8 E8 A4 DF 88 D0 E4 8E 42 17 A9 00 8D 41
00D0 17 A6 E9 CA D0 CC 20 40 1F 4C 00 01 00 00 00 05

0100 20 6A 1F C9 15 F0 15 85 E6 C9 00 F0 0F 85 EC 18
0110 65 EC 85 EB 65 EB 85 EB 65 EB 85 EA 60 A2 04 B5
0120 E0 95 E1 CA 10 F9 60 A5 EE A2 FF E8 E0 30 F0 0B
0130 DD 40 01 D0 F6 BD 70 01 85 E0 60 A9 08 85 E0 60
0140 05 18 1A 0C 02 12 0E 10 04 17 0D 14 07 06 0F 16
0150 1D 0A 08 03 09 11 0B 19 1B 1C 2F 27 23 21 20 30
0160 38 3C 3E 3F 4C 31 61 32 55 5E 52 36 6D 2A 45 22
0170 77 7C 39 5E 79 71 3D 74 04 1F 78 38 37 54 5C 73
0180 E7 50 6D 31 3E 1C 7E 76 6E 5B 06 5B 4F 66 6D 7D
0190 07 7F 6F 3F 53 40 40 52 0C 02 22 61 43 41 49 1C
```

Zero-Page-Adressen

EE Zeichenpuffer	E9 Zeitbasis-Zähler
ED Zähler für die Elementlänge	E8 Zähler f. Pausenlänge
EC Halbe Punktlänge	E7 Störimpuls-Zähler
EB Zwei Punktlängen	E6 Automatik-Flag
EA Vier Punktlängen	DF Save Y (f. Display)

Der angezeigte Text steht im Adressenbereich 00E0...00E5.

werden, was aber noch schlechtere Ergebnisse bringen würde. Bei einer Pause von mehr als zwei Punktlängen wurde ein vollständiges Zeichen empfangen und über eine Umwandlungstabelle für die Morsezeichen (0140...016F) und das Siebensegment-Alphabet (0170...019F) in das KIM-Display geschoben. Dabei wird ein ASCII-ähnlicher Zwischencode verwendet.

Tritt eine Pause von mehr als vier Punktlängen auf, so wird sie als Wortzwischenraum interpretiert, und in das Display wird zusätzlich eine Leerstelle geschoben.

Manuelle und automatische Geschwindigkeitswahl

Wie schon erwähnt, kann sich der KIM-1 automatisch an die Geschwindigkeit der empfangenen Morsezeichen anpassen. Verständlicherweise funktioniert das aber nur in gewissen Grenzen und bei relativ wenig gestörten Signalen. Aus diesem Grunde steht zusätzlich eine manuelle Geschwindigkeitswahl zur Verfügung, die ebenfalls über die KIM-Tastatur erfolgt. Die Taste 1 ergibt das schnellste Tempo, nach F hin wird es langsamer. Beim Drücken der Taste 0 wird die Automatik eingeschaltet.

Einen Anhaltspunkt für die richtige Geschwindigkeit bilden die in der Anzeige erscheinenden Zwischenräume. Erscheint nach jedem Buchstaben ein Zwischenraum, so läuft der KIM zu schnell; läuft er dagegen zu langsam, so „kleben" auch die Worte aneinander. Im praktischen Betrieb hat sich die manuelle Geschwindigkeitswahl als wesentlich günstiger erwiesen. Erstens schwanken die Geschwindigkeiten der empfangenen Stationen meist nicht allzu sehr, und zweitens kann die Automatik bei anhaltenden Störungen doch außer Tritt geraten.

Noch ein Tip: Speichert man das Programm auf Band, so ist es ganz nützlich, wenn man ab der Zelle 00E0 die Daten 53 5C 73 37 79 31 einschreibt. Beim Programmstart erscheint dann im Display zunächst die Frage „Tempo?", um auf die Notwendigkeit der Geschwindigkeitswahl hinzuweisen, da nach jedem Programmstart der KIM vorerst auf einem sehr langsamen Tempo steht.

Gerhard Kirchberger
Herwig Feichtinger

RAM-Testprogramm für den Z-80-KIT

Oft erweitert man den RAM-Bereich seines Mikroprozessors, weiß aber nicht, ob die zugekauften RAM-ICs in Ordnung sind. Die neuen Speicherplätze einzeln zu überprüfen, wäre zu mühsam und zeitaufwendig. Das hier beschriebene Programm führt diese Überprüfung automatisch und in kurzer Zeit durch (6 s für 1 KByte). Das *Bild* zeigt das Flußdiagramm.

Die Anfangsadresse des zu überprüfenden Speicherbereiches muß in die Register H und L, die Endadresse in die Register D und E eingegeben werden, wobei beachtet werden sollte, daß stets DE ≧ HL ist; sonst läuft das Programm über die Adresse 0000 und löscht sich bei Erreichen seiner eigenen Adresse selbst.

Nach dem Start lädt das Programm den durch HL adressierten Speicherplatz mit den Daten 00 bis FF und überprüft nach jedem Laden, ob der Speicherplatz auch diese Daten enthält. Wird diese Überprüfung immer bejaht, so wird HL um 1 erhöht, und das Programm überprüft nun diesen nächsten Speicherplatz. Das geschieht so lange, bis der Wert von HL den Wert von DE erreicht hat. Danach springt das Programm in die Anzeigeroutine des Z-80-KIT; es ist mit der Anzeige der Endadresse beendet.

Wird die Überprüfung des Speicherplatzes verneint, d. h. enthält der Speicherplatz nicht die Daten, mit denen er vorher geladen worden ist, so springt das Programm ebenfalls in die Anzeigeroutine, und die Adresse des „defekten" Speicherplatzes wird im Adressenanzeigeteil angezeigt.

Jörg Ullrich

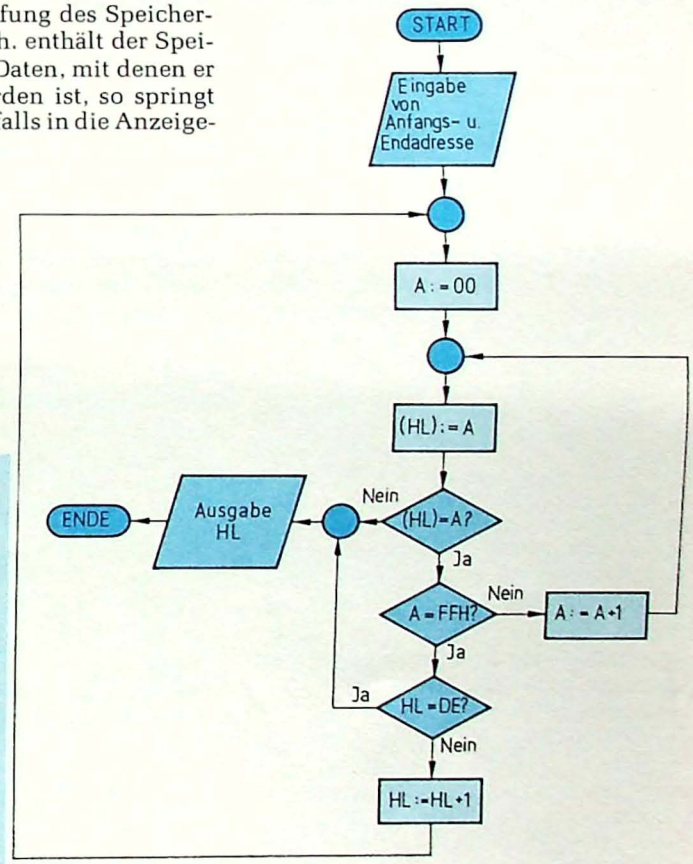
Anmerkung der Redaktion:

Über RAM-Testmethoden gibt es die unterschiedlichsten Auffassungen und Ideologien. Fest steht, daß 95 % aller Fehler schon dadurch gefunden werden können, indem jede Speicherzelle wechselweise mit 00 und FF beaufschlagt wird. Die hier vorgestellte Methode findet bestimmt nicht alle übrigen möglichen Fehler, stellt aber wegen der geringen erforderlichen Prüfzeit eine universell verwendbare Version dar.

Maschinencode für den Z-80-KIT

Stack-Pointer 3CFFH
Startadresse 3C50H

Adresse	Code
3C50	3E00
3C52	77
3C53	BE
3C54	C20D01
3C57	FEFF
3C59	2803
3C5B	3C
3C5C	18F4
3C5E	7C
3C5F	BA
3C60	2005
3C62	7D
3C63	BB
3C64	CA0D01
3C67	23
3C68	18E6



Flußdiagramm des RAM-Tests. Das Registerpaar HL enthält die Anfangsadresse des zu prüfenden Speicherbereichs. DE beinhaltet die Endadresse. Mit A ist der Akkumulator und mit (HL) der durch HL adressierte Speicherplatz bezeichnet

Will man einem Mikrocomputer das Rechnen beibringen, so stellen die vier Grundrechenarten bekanntlich die größten Probleme dar. Die folgenden Routinen befassen sich daher auch nur mit Addieren und Subtrahieren, Multiplizieren und Dividieren.

Michael Konz

6502-Rechenroutinen

Das nachfolgend beschriebene Programm wurde für den AIM-65 geschrieben, der gewissermaßen als Nachfolger des KIM bezeichnet werden kann. Natürlich laufen die Routinen auch auf diesem, wenn geringfügige Änderungen, auf die später noch eingegangen werden wird, vorgenommen werden. Eine alphanumerische Ein- und Ausgabemöglichkeit wäre jedoch nützlich, wenn auch nicht unbedingt erforderlich.

Die Genauigkeit der Routinen läßt sich durchaus mit der guter Taschenrechner vergleichen. In den meisten Fällen liegt sie sogar höher. Es können Zahlen im Bereich von $1 \cdot 10^{-99}$ bis $9{,}999999999 \cdot 10^{99}$ verarbeitet werden. Intern wird mit einer zwölfstelligen Mantisse gearbeitet. Diese Stellen können bei entsprechender Parametereingabe auch alle angezeigt werden. Dennoch ist solch eine Programmierung nicht zu empfehlen, will man nicht auf Schutzstellen verzichten.

Bevor die einzelnen Routinen nun besprochen werden, soll eine Auflistung aller dem Anwender zur Verfügung stehenden Unterprogramme gegeben werden.

0200
Der Register-Pointer wird abhängig vom Inhalt der beiden Indexregister gesetzt.

0233
Ein Register wird in ein anderes kopiert. Angabe von Ursprungs- und Zielregister erfolgt durch die beiden Indexregister der CPU.

0240
Das x-Register wird in das y-Register kopiert.

0246
Das y-Register wird in das x-Register kopiert.

0259
Das x-Register wird mit dem y-Register vertauscht.

0269
Das durch das Indexregister X angegebene Register wird gelöscht.

027C
Exponent und Vorzeichen des durch den Index X angegebenen Registers werden gelöscht.

028D
Der Exponent im x-Register wird um 1 erhöht. Tritt ein Überlauf auf, so erfolgt eine Fehlermeldung und Rückkehr zum Monitor.

02AB
Der Exponent im x-Register wird um 1 erniedrigt; sonst wie 028D.

02BB
Die Mantisse im x-Register wird um 1 nach links verschoben, was einer Multiplikation mit 10 gleichkommt. Ein dabei auftretender Übertrag bleibt unberücksichtigt.

02C7
Die Zahl im x-Register wird in die Exponentialform gebracht.

02ED
Die Exponenten der Zahlen in x und y werden gleichgesetzt.

0354
Die Mantisse von x wird um 1 nach rechts verschoben, was einer Division durch zehn gleichkommt. Die letzte Ziffer geht verloren, der Exponent bleibt unverändert.

0369
Wie 0354, jedoch für das y-Register.

037E
Der Inhalt des x-Registers wird ausgegeben.

0437
Die Inhalte des x- und des y-Registers werden verglichen, das Ergebnis des Vergleiches an der Zero-Page-Adresse F2 gespeichert.

044C
Wie 0437, es werden aber nur die absoluten Beträge verglichen.

0464
Wie 0437, es werden jedoch nur die Mantissen verglichen.

04B8
Die Inhalte von x und y werden addiert, das Ergebnis im x-Register gespeichert.

0518
Die Inhalte von x und y werden subtrahiert (x–y), das Ergebnis im x-Register gespeichert.

0523
Eingabe einer Zahl in das x-Register.

05CF
Die Inhalte von x und y werden multipliziert, das Ergebnis im x-Register gespeichert.

069A
Der Inhalt von x wird durch den Inhalt von y dividiert, das Ergebnis in x gespeichert.

0766
Ausgabe des Textes „UNVERSTAENDLICHE EINGABE".

0768
Ausgabe eines beliebigen alphanumerischen Textes.

077A
Ausgabe des Textes „RECHENKAPAZITAET UEBERSCHRITTEN" und Rückkehr zum Monitor.

Um Verwirrungen zu vermeiden, muß zunächst einmal auf die Register eingegangen werden. Die von den Routinen benutzten Register dienen zur Aufnahme der Daten und belelgen je 16 Byte im Speicher. Sie dürfen nicht mit den beiden Indexregistern der CPU verwechselt werden.

In der Zero-Page stehen insgesamt 15 Register zur Verfügung. Davon werden neun von den Routinen belegt, die übrigen kann der Programmierer zur eigenen Datenspeicherung verwenden. Die beiden ersten Register erhielten den Namen x und y. Sie sind an den Adressen 0000-000F und 0010-001F zu finden. Weiterhin kann der Anwender ab einer von ihm bestimmbaren Page beliebig viele zusätzliche Register festlegen. Dies geschieht durch Schreiben der Zero-Page-Adresse F4. Der Wert 0A bezeichnet dann den Speicherraum ab der Adresse 0A00. Dieser Parameter ist vor dem ersten Aufruf einer Transfer-Routine einzugeben. Natürlich muß dabei darauf geachtet werden, daß die Daten nicht das Programm zerstören,

```
0200 8A 20 18 02 85 FB 20 2D 02 85 FA 98 AA 20 18 02
0210 85 FD 20 2D 02 85 FC 60 29 F0 C9 F0 D0 04 A9 00
0220 F0 0A 8A 4A 4A 4A 4A D8 18 65 F4 F8 60 8A 0A 0A
0230 0A 0A 60 20 00 02 A0 0F B1 FA 91 FC 88 10 F9 60
0240 A2 F0 A0 F1 30 ED A2 F1 A0 F0 30 E7 A2 F0 A0 F2
0250 20 33 02 A2 F1 A0 F3 30 DA A2 F1 A0 FD 20 33 02
0260 20 40 02 A2 FD A0 F0 30 CA 20 00 02 A0 0F A9 00
0270 91 FA 88 10 FB A9 01 A0 0D 91 FA 60 20 00 02 A0
0280 0C A9 00 91 FA C8 C8 91 FA C8 91 FA 60 F8 A5 0E
0290 30 0D A5 0F 18 69 01 85 0F 90 03 4C 7A 07 60 A5
02A0 0F 38 E9 01 85 0F D0 F6 85 0E 60 F8 A5 0F D0 05
02B0 E6 0F C6 0E 60 24 0E 30 DB 10 E6 A2 00 B5 01 95
02C0 00 E8 E0 0B 90 F7 60 F8 A0 00 A5 00 F0 0F A5 0D
02D0 C9 01 D0 01 60 C6 0D 20 8D 02 B8 50 F1 20 BB 02
02E0 20 AB 02 C8 C0 0C 90 E2 A2 F0 4C 7C 02 A5 0E C5
02F0 1E F0 1F 24 0E 10 0D A5 0F F0 44 20 54 03 20 8D
0300 02 B8 50 F3 A5 1F D0 04 E6 1E 10 2A 20 62 03 B8
0310 50 F2 A5 0F C5 1F D0 01 60 24 0E 10 17 90 0C 20
0320 4B 03 20 54 03 20 8D 02 B8 50 F4 20 4B 03 20 62
0330 03 B8 50 F7 90 09 20 4B 03 20 77 03 B8 50 F7 20
0340 4B 03 20 54 03 20 8D 02 B8 50 F4 A5 0F C5 1F D0
0350 02 68 68 60 A2 0A B5 00 95 01 CA 10 F9 A9 00 85
0360 00 60 A5 1F 38 E9 01 85 1F A2 0A B5 10 95 11 CA
0370 10 F9 A9 00 85 10 60 A5 1F 18 69 01 90 E9 20 44
0380 EB A9 03 85 F0 A9 00 85 F1 A2 F0 A0 FD 20 33 02
0390 A5 F1 30 23 A5 0F C9 10 B0 1D 24 0E 30 0C A5 0F
03A0 F0 15 E6 0D 20 AB 02 B8 50 F4 A5 0F F0 09 20 54
03B0 03 20 8D 02 B8 50 F3 D8 18 A5 0D 65 F0 F8 C9 0C
03C0 90 02 A9 0B AA 86 F0 B5 00 C9 05 90 24 CA EA 18
03D0 B5 00 69 01 48 0A 0A 0A 0A 68 29 0F 95 00 90 11
03E0 CA 10 EC E6 0D 20 54 03 E6 00 A5 F1 10 03 20 C7
03F0 02 A2 00 A5 0C 10 04 A9 2D 10 02 A9 20 20 7A E9
0400 B5 00 20 51 EA E8 E4 F0 B0 08 E4 0D D0 F2 A9 2E
0410 10 EB A5 0F F0 1A 20 3E E8 A9 45 20 7A E9 A5 0E
0420 10 04 A9 2D 10 02 A9 2B 20 7A E9 A5 0F 20 46 EA
0430 A2 FD A0 F0 4C 33 02 A5 0C C5 1C 90 07 F0 09 A9
0440 80 85 F2 60 A9 40 10 F9 A5 0C 30 37 A5 0E C5 1E
0450 90 F2 F0 02 B0 E9 A5 0E 30 1F A5 0F 30 1B 90 DF
0460 F0 02 B0 E0 A2 00 B5 00 D5 10 90 D3 F0 02 B0 D4
0470 E8 E0 0C 90 F1 A9 00 F0 C8 A5 0F C5 1F 90 C5 F0
0480 E3 B0 BC A5 0E C5 1E 90 B6 F0 02 B0 B7 A5 0E 30
0490 0A A5 0F C5 1F 90 AD F0 0C B0 A4 A5 0F C5 1F 90
04A0 9E F0 02 B0 9F A2 00 B5 00 D5 10 90 97 F0 02 B0
04B0 8E E8 E0 0C 90 F1 B0 BD 20 ED 02 A5 0C C5 1C D0
04C0 21 A2 0B 18 B5 00 75 10 48 0A 0A 0A 0A 68 29 0F
04D0 95 00 CA 10 EF 90 07 20 54 03 E6 0D E6 00 20 C7
04E0 02 60 20 64 04 A5 0C 10 08 24 F2 70 1F E6 0C 10
04F0 06 24 F2 70 17 C6 0C 38 A2 0B B5 10 F5 00 29 0F
0500 95 00 CA 10 F5 B0 D7 20 54 03 F0 D2 38 A2 0B 20
0510 B4 05 EA CA 10 F9 30 ED 20 ED 02 A5 0C C5 1C D0
0520 A0 F0 BF A2 0F A9 00 95 00 CA 10 FB 20 F0 E9 4E
0530 11 A4 A0 00 20 5F E9 C8 C9 0D D0 F8 AE 15 A4 9D
0540 38 A4 0E 11 A4 A2 00 A0 00 B9 38 A4 C9 0D F0 2F
0550 C9 2D D0 05 C6 0C C8 10 F0 C9 2E D0 08 86 0D A5
0560 0D F0 17 D0 F1 C9 45 F0 23 20 7D EA B0 0C C9 0A
0570 B0 08 95 00 E8 C8 E0 0B 90 CF 20 66 07 50 A4 A5
0580 0D D0 02 86 0D 20 C7 02 4C 7E 03 EA C8 B9 38 A4
0590 C9 2D D0 06 C6 0E C8 B9 38 A4 C9 3A B0 DC 20 84
05A0 EA B0 D7 C8 B9 38 A4 C9 3A B0 CF 20 84 EA B0 CA
05B0 85 0F 90 CB B5 00 F5 10 29 0F 95 00 60 A5 FF C6
05C0 FE 30 09 C6 FE 30 07 18 65 FF 90 F7 A9 00 60 20
05D0 4C 02 A9 00 85 FA 20 6C 02 A9 10 85 FA 20 6C 02
05E0 A2 00 D8 A9 0B 86 F8 38 E5 F8 A8 F8 B5 30 F0 50
05F0 85 FF A2 0B B9 20 00 85 FE 20 BD 05 48 29 0F 18
0600 75 10 48 0A 0A 0A 0A 68 29 0F 95 10 68 08 CA 10
0610 12 48 20 69 03 68 4A 4A 4A 4A 28 65 10 85 10 E6
0620 1D 10 0C 4A 4A 4A 4A 28 75 10 95 10 88 10 C5 20
0630 59 02 20 C7 02 20 59 02 20 B8 04 A2 F1 20 69 02
0640 A6 F8 E8 E0 0C 90 9B 20 4D 06 4C C7 02 A5 0F F0
0650 04 C6 0F E6 0D A5 2C C5 3C F0 02 C6 0C A5 2E C5
0660 3E D0 0F 85 0E 18 A5 2F 65 3F 90 03 4C 7A 07 85
0670 0F 60 A5 2F C5 3F 24 2E 10 12 90 08 38 E5 3F 85
0680 0F C6 0E 60 A5 3F 38 E5 2F 85 0F 60 90 06 20 7C
0690 06 E6 0E 60 20 84 06 C6 0E 60 A2 F0 A0 F4 20 33
06A0 02 A2 F1 A0 F5 20 33 02 A2 F1 20 7C 02 A2 F0 20
06B0 69 02 20 37 04 A5 F2 D0 0B A2 FF 9A A2 BE 20 68
06C0 07 4C 82 E1 A2 F4 A0 F0 20 33 02 A2 F0 20 7C 02
06D0 A2 F0 A0 F7 20 33 02 A2 F6 20 69 02 A2 00 86 F7
06E0 86 F9 20 18 05 A5 0C 30 04 E6 F7 10 F5 A5 F7 A6
06F0 F9 95 60 E8 E0 0C B0 3F 86 F9 48 20 46 02 20 C7
0700 02 A2 F1 20 69 02 68 85 10 20 CF 05 20 40 02 A2
0710 F7 A0 F0 20 33 02 20 18 05 A2 F5 A0 F1 20 33 02
0720 A2 F1 20 7C 02 E6 0D 20 C7 02 A2 F0 A0 F7 20 33
0730 02 A2 00 86 F7 F0 AB A2 F6 A0 F0 20 33 02 20 44
0740 07 4C C7 02 A2 F4 A0 F2 20 33 02 A2 F5 A0 F3 20
0750 33 02 A5 2C C5 3C F0 02 C6 0C A5 2E C5 3E D0 03
0760 4C 72 06 4C 63 06 A2 85 20 F0 E9 BD 00 07 C9 02
0770 F0 06 20 7A E9 E8 D0 F3 B8 60 A2 9E 20 68 07 A2
0780 FF 9A 4C 82 E1 55 4E 56 45 52 53 54 41 45 4E 44
0790 4C 49 43 48 45 20 45 49 4E 47 41 42 45 02 52 45
07A0 43 48 45 4E 4B 41 50 41 5A 49 54 41 45 54 20 55
07B0 45 42 45 52 53 43 48 52 49 54 54 45 4E 02 52 45
07C0 53 55 4C 54 41 54 20 55 4E 45 4E 44 4C 49 43 48
07D0 02 03
```

Bild 1. Dieses hexadezimale Listing enthält alle AIM-65-Rechenroutinen

das sich an den Speicherplätzen 0200-07D0 befindet.

Innerhalb der Register sind die Daten folgendermaßen angeordnet:
XXX0-XXXB: 12stellige Mantisse

XXXC: Vorzeichen der Mantisse (00: positiv, FF: negativ)
XXXD: Stellung des Dezimalpunktes
XXXE: Vorzeichen des Exponenten
XXXF: Exponent

Nun zu den einzelnen Routinen. Zum Transfer eines Registers in ein anderes wird die Routine ab der Adresse 0233 verwendet. Das Ursprungsregister wird durch eine hexadezimale Ziffer angegeben, die vor dem Aufruf der Routine im Indexregister X gespeichert werden muß. Das Zielregister wird genauso im Index Y gespeichert. Dabei bezeichnen die Zahlen FX Register in der Zero-Page. F0 ist das x-Register, F1 das y-Register. Alle anderen Register beziehen sich auf die Angabe in F4. Ist dieser Parameter z. B. 0A, dann bezeichnet das Register 00 die 16 Byte an den Adressen 0A00...0A0F; 01 ist bei 0A10...0A1F zu finden usw.

Zu den Registern wäre noch zu sagen, daß das Register FF, das in der Zero-Page an den Adressen F0...FF stehen würde, nicht benutzt werden kann, da die FX-Adressen für den Programmablauf wichtige Parameter enthalten.

Für die Routinen 0269 und 027C gilt das oben Gesagte für die Adressierung der Register. Es wird jedoch nur ein Wert im Index X benötigt.

Die Routine 0200 schließlich dient dazu, den Pointer auf die angegeben Register zu setzten. Dieser Pointer befindet sich an den Adressen FA und FB bzw. FC und FD. Normalerweise wird diese Routine nicht benötigt, da eine Transferroutine ihren Aufruf beinhaltet.

Die Routine 02C7 gehört zu den wichtigsten überhaupt. Sie sorgt dafür, daß alle Daten nach jeder Eingabe und nach jeder Rechenoperation immer in Exponentialform vorliegen:
$1{,}23456789012 \cdot 10^{99}$

Die Routine 02ED ist nicht weniger wichtig. Sie gleicht die Exponenten im x- und y-Register aneinander an. Dazu ein Beispiel:

Im x-Register steht die Zahl $1{,}23 \cdot 10^{08}$,
im y-Register die Zahl $5{,}67 \cdot 10^{05}$.

Nach Aufruf der Routine steht die Zahl im x-Register unverändert, im y-Register steht jedoch $0{,}00567 \cdot 10^{08}$. Es ist sofort zu sehen, daß der Wert derselbe ist wie zuvor.

Zur Ausgabe von Daten stehen zwei Formate zur Verfügung. Wird die Routine 037E aufgerufen, so erfolgt die Anzeige mit Festkomma, das auf drei Stellen gesetzt ist, soll die Anzeige im Exponentialformat erfolgen, so müssen zuvor die Zero-Page-Adressen F0 und F1 beschrieben werden. Normalerweise stehen hier die Werte 03 und 00. Für das Exponentialformat muß der Wert von F1 zu FF geändert werden. An F0 muß dann die Zahl der Nachkommastellen geschrieben werden. Die Startadresse lautet dann 0389. In beiden Fällen findet eine automatische 5/4-Rundung statt. Vorsicht: Beim Aufruf der Routine wird das Register FD überschrieben.

Die nächsten drei Routinen sind eigentlich eine einzige mit drei verschiedenen Startadressen. Wie bereits gesagt, dienen sie zum Vergleich des x- und des y-Registers. Das Resultat des Vergleiches hat folgende Form:

Ist x kleiner als y, dann wird Bit 7 von F2 gesetzt.

```
0800 A2 FC A0 F0 20 33 02 A2 F9 A0 F1 20 33 02 20 CF
0810 05 A2 F0 A0 FC 20 33 02 A2 F0 20 69 02 A2 FA A0
0820 F1 4C 33 02 20 C7 02 20 59 02 20 CF 05 A2 F0 A0
0830 FA 20 33 02 A9 00 85 AC 20 40 02 A2 FC A0 F0 20
0840 33 02 20 9A 06 20 40 02 A2 F8 A0 F0 20 33 02 20
0850 B8 04 A2 F0 A0 F8 20 33 02 4C 00 08 20 23 05 A2
0860 F0 A0 F8 20 33 02 20 40 02 20 CF 05 A2 F0 A0 F9
0870 20 33 02 A2 FA 20 69 02 E6 A0 A2 F8 A0 FC 20 33
0880 02 A9 00 85 F6 20 00 08 20 99 08 C6 0C 20 24 08
0890 20 99 08 20 24 08 4C 88 08 A6 F6 E0 15 90 0F 68
08A0 68 A2 F8 A0 F0 20 33 02 20 F0 E9 4C 7E 03 A0 00
08B0 BD C3 08 99 00 00 E8 C8 C0 03 90 F4 A9 02 85 0F
08C0 86 F6 60 00 00 06 00 02 00 00 04 02 00 07 02 01
08D0 01 00 01 05 06 02 01 00 20 5C 08 20 5F E9 4C D8
08E0 08
```

Bild 2. Berechnung des Sinus. Startadresse ist 08D8

Ist x größer als y, dann wird Bit 6 von F2 gesetzt.

Ist x gleich y, dann wird F2 mit dem Wert 00 geschrieben.

Das Ergebnis des Vergleiches kann also leicht mit der Instruktion BIT F2 abgefragt werden, gefolgt von BMI, BPL, BVC oder BVS, beziehungsweise von der Befehlsfolge LDA F2; BEQ/BNE.

Die Routine 04B8 ist die erste wirkliche Rechenroutine. 0523 dient zur Übernahme von Daten ins x-Register. Bei der Eingabe ist folgendes zu beachten: Das Vorzeichen ist immer vor der Zahl einzugeben (Im Gegensatz zu den meisten Taschenrechnern). Die Exponenteneingabe wird mit einem E begonnen. Der Exponent muß immer zweistellig eingegeben werden. Die Eingabe –123E-8 würde zu einer Fehlermeldung führen. Richtig muß es heißen: –123E-08.

Eine fehlerhafte Eingabe kann durch Betätigen der DEL-Taste rückgängig gemacht werden, solange nicht RETURN gedrückt wurde. Die zulässige Stellenanzahl kann durch den Wert an der Adresse 0577 bestimmt werden. Eine Stellenzahl von 10 ist vorgesehen. Mehr Ziffern führen nach RETURN zu einer Fehlermeldung. Zu beachten ist, daß nach einer Fehlermeldung die Eingabe automatisch wiederholt wird, bis eine korrekte Eingabe erfolgt.

Sehr nützlich ist die Routine 0768, die dem Anwender erlaubt, eigene Erläuterungen zu Rechenoperationen ausgeben zu lassen. Es stehen hier noch 47 Byte für beliebige ASCII-Zeichen zur Verfügung. Diese werden ab der Adresse 07D1 bis maximal 07FF gespeichert. Jeder Text wird mit CNTRL B (Code 02) beendet. Vor jedem Aufruf muß die Startadresse für den Text im Indexregister X gespeichert werden, d. h. das niederwertige Adreßbyte. Werden die drei Fehlermeldungen der Routinen vom Anwender abgekürzt, so steht ein größerer Raum für eigene Texte zur Verfügung. Dazu die Adressen:

06BD: Fehlermeldung bei Division durch 0

0767: Fehlermeldung bei fehlerhafter Eingabe

077B: Fehlermeldung bei Überschreiten der Rechenkapazität

Der Textbuffer beginnt bei der Adresse 0785.

Soviel zu den Routinen. Ein kleines Beispiel soll zeigen, wie aus ihnen komplexe Funktionen zusammengesetzt werden können. *Bild 2* zeigt ein Programm zur Berechnung des Sinus. Die Eingabe erfolgt im Bogenmaß, Startadresse ist 08D8.

Wie schon gesagt, sind einige Änderungen erforderlich, soll das Programm auf einem anderen System laufen. Die größten Schwierigkeiten dürften dabei bei der Eingabe-Routine auftreten, da der Display-Buffer des AIM als Zwischenspeicher benutzt wird. Nur so ist die Benutzung der DELETE-Funktion für Korrekturen möglich. Der Anwender muß hier ggf. nach einer gleichwertigen Lösung suchen oder auf die Korrekturmöglichkeit verzichten.

Ferner finden folgende Routinen des AIM-Monitors Verwendung:

E97A

Ausgabe eines ASCII-Zeichens im Akku.

EA51

Ausgabe der vier niederwertigen Bits des Akku als ASCII-Zeichen.

EA46

Wie EA51,. jedoch werden zuvor die vier höherwertigen Bits ausgegeben.

EA7D

Ein ASCII-Zeichen im Akku wird in eine hexadezimale Ziffer von 0-F verwandelt.

EA84

Wie EA7D, jedoch werden bei zweimaligem Aufruf zwei ASCII-Zeichen in ein Byte gepackt. In beiden Fällen wird das Carry-Flag gesetzt, wenn es sich bei dem Zeichen um keine hexadezimale Ziffer handelt.

E9F0

Ausgabe einer CRLF-Folge bzw. Löschen des Display.

E95F

Eingabe eines ASCII-Zeichens von der Tastatur in den Akku.

Abschließend noch einige Worte zur Rechengeschwindigkeit. Hier die Werte für die verschiedenen Operationen:

Addition: 0,5...3,5 ms
Subtraktion: 0,5...3,5 ms
Multiplikation: 4,5...60 ms
Division: 140...180 ms

Die Werte gelten für eine Taktfrequenz von 1 MHz. Diese Geschwindigkeiten erscheinen recht langsam, vergleicht man sie mit denen guter programmierbarer Taschenrechner. Die Überlegenheit des Mikrocomputers zeigt sich erst in langen Programmen, da zwischen der Ausführung zweier Operationen nur wenig Mikrosekunden vergehen. Dadurch wird die relativ geringe Rechengeschwindigkeit mehr als ausgeglichen.

Als letztes sollte noch gesagt werden, daß, wenn der Anwender innerhalb seines eigenen Programmes den Dezimalmodus mit dem Befehl CLD aufhebt, vor Aufruf der nächsten Rechenroutine unbedingt der gegenteilige Befehl, nämlich SED, folgen muß, sollen die Routinen einwandfrei arbeiten. ■

Textausgabe in Maschinensprache

Im FUNKSCHAU-Sonderheft HOBBYCOMPUTER 2 beschrieb E. H. Franke einen Weg, wie ein schwerwiegender Nachteil der Maschinensprache behoben werden kann.

```
PRTSTR  POP HL
        LD A,(HL)
        INC HL
        PUSH HL
        OR A
        RET Z
        RST 10H
        JR PRTSTR
```

Unterprogramm zur Textstring-Ausgabe. Der Text ist als ASCII-Folge direkt an den Aufruf dieses Unterprogrammes zu hängen und muß mit hex 00 (NOP) abgeschlossen werden

Sein 6502-Assemblerprogramm erlaubt es, den Text als ASCII-Bytefolge direkt hinter einen Unterprogrammaufruf zu schreiben, und das Programm wird hinter dem mit NOP abgeschlossenen Text fortgesetzt.

Die vorliegende Version für den TRS-80 unter Level-1-Basic funktioniert nach dem gleichen Prinzip.

Ing. (grad.) Rolf-Fr. Matthaei

Bereits in Heft 21/1978 der FUNKSCHAU wurde für 6502-Mikrocomputer ein Disassembler veröffentlicht. Dieser dient dazu, in Maschinencode vorliegende Programme wieder in den symbolischen Assemblercode zurückzuwandeln. Der folgende Beitrag beschreibt eine verbesserte Version, die auch die Operationscodes ausgibt.

Martin Görlitz

6502-Disassembler

Bereits der FUNKSCHAU-Disassembler kam mit einem relativ geringen Speicherplatz aus, da er nicht über große Tabellen alle möglichen Instruktionen abprüft und entsprechend umwandelt, sondern nach einem ausgetüftelten Verfahren gewisse Gruppen von Befehlen jeweils abtrennt und umsetzt. Da nicht alle der 256 möglichen Bitkombinationen einer 8-bit-Instruktion tatsächlich einen Maschinenbefehl bedeuten, kann hiermit die Prüfung sehr verkürzt werden. Allerdings muß man beim Aufstellen solcher Tabellen sehr sorgfältig vorgehen.

Eine Gruppe von Befehlen konnte der FUNKSCHAU-Disassembler nicht verstehen, und zwar ROR-(rotate right)-Befehle. Die abgedruckten Tabellen sind diesbezüglich korrigiert. Nunmehr sind alle Instruktionen wieder rückassemblierbar.

Bei einem Assemblerlisting ist natürlich nicht nur der mnemonische Code und die Adressierungsart, sondern auch der Maschinencode interessant. Manchmal läuft der Disassembler bei der Arbeit aus Versehen in eine Tabelle hinein, und dann kann man meist nur am Maschinencode prüfen, daß hier z.B. eine ASCII-Textzeile im Programm drinsteht. So etwas sollte nach Meinung aller Programmexperten unbedingt vermieden werden, aber wer schreibt schon alle Programme an einem Tag, und wenn man zu vorhandenen Programmen ein weiteres hinzufügt, kommen fast immer auch Tabellen hinzu. Schon ist das Durcheinander komplett.

An dieser Stelle sei ein Hinweis erlaubt: In der Zeitschrift Micromag wurde vorgeschlagen, am Beginn jeder Tabelle einen „Merker" anzufügen, der keine Maschineninstruktion sein darf (beim 6502 z. B., wenn die beiden kleinsten Bits 1 sind), und so den Beginn der Tabelle und deren Länge anzugeben. Der Disassembler könnte somit auf das automatische Überspringen von Tabellen ausgeweitet werden; die

```
COCOS - Disassembler
====================

Startadresse:       D4F3, wenn jeweils nach einer Bildschirmseite
                          angehalten werden soll (weiter mit Leertaste)
                    D4F7, wenn kontinuierlich auszudrucken ist. Hierzu ist
                          EB mit einem negativen Wert zu laden.
Ausgabepuffer:      29 Bytes am Ende von PAGE 3
Ausgaberoutine:     D610  CRLF (=KIM 1E2F)
                    D618  OUTCHAR (=KIM 1EA0)
Aufruf:             mit Startadresse in 17F5-F6 und Stopadresse in 17F7-F8

D4F3   A9 10      LDA #$10
D4F5   85 EB      STA   EB
D4F7   AD F5 17   LDA 17F5
D4FA   85 FA      STA   FA
D4FC   AD F6 17   LDA 17F6
D4FF   85 FB      STA   FB
D501   A2 1D      LDX #$1D
D503   A9 20      LDA #$20
D505   9D E2 03   STA 03E2,X
D508   CA         DEX
D509   D0 F8      BNE D503
D50B   A5 FB      LDA   FB
D50D   20 AB D6   JSR D6AB
D510   8C E3 03   STY 03E3
D513   8D E4 03   STA 03E4
D516   A5 FA      LDA   FA
D518   20 AB D6   JSR D6AB
D51B   8C E5 03   STY 03E5
D51E   8D E6 03   STA 03E6
D521   A2 00      LDX #$00
D523   A0 00      LDY #$00
D525   B1 FA      LDA  (FA),Y
D527   3D C8 D6   AND D6C8,X
D52A   5D D8 D6   EOR D6D8,X
D52D   F0 0D      BEQ D53C
D52F   E8         INX
D530   E0 10      CPX #$10
D532   D0 EF      BNE D523
D534   A9 3F      LDA #$3F
D536   8D FF 03   STA 03FF
D539   4C 29 19   JMP 1929
D53C   BD E0 D7   LDA D7E0,X
D53F   85 F4      STA   F4
D541   A9 D5      LDA #$D5
D543   85 F0      STA   F0
D545   BD F0 D7   LDA D7F0,X
D548   85 EF      STA   EF
D54A   A6 F4      LDX   F4
D54C   CA         DEX
D54D   F0 1D      BEQ D56C
D54F   CA         DEX
D550   F0 0D      BEQ D55F
D552   A0 02      LDY #$02
D554   B1 FA      LDA  (FA),Y
D556   20 AB D6   JSR D6AB
D559   8C F9 03   STY 03F9
D55C   8D FA 03   STA 03FA
D55F   A0 01      LDY #$01
D561   B1 FA      LDA  (FA),Y
D563   20 AB D6   JSR D6AB
D566   8C FB 03   STY 03FB
D569   8D FC 03   STA 03FC
D56C   6C EF 00   JMP(00EF)
D56F   A9 23      LDA #$23
D571   8D F9 03   STA 03F9
D574   A9 24      LDA #$24
D576   D0 02      BNE D57A
D578   A9 41      LDA #$41
D57A   D0 1F      BNE D59B
D57C   A9 29      LDA #$29
D57E   8D FD 03   STA 03FD
D581   A9 2C      LDA #$2C
D583   8D FE 03   STA 03FE
D586   A9 59      LDA #$59
D588   D0 0C      BNE D596
D58A   A9 2C      LDA #$2C
D58C   8D FD 03   STA 03FD
D58F   A9 58      LDA #$58
D591   8D FE 03   STA 03FE
D594   A9 29      LDA #$29
D596   8D FF 03   STA 03FF
D599   A9 28      LDA #$28
D59B   8D FA 03   STA 03FA
D59E   D0 17      BNE D5B7
D5A0   A9 58      LDA #$58
D5A2   D0 02      BNE D5A6
D5A4   A9 59      LDA #$59
D5A6   8D FE 03   STA 03FE
D5A9   A9 2C      LDA #$2C
D5AB   D0 07      BNE D5B4
D5AD   A9 28      LDA #$28
D5AF   8D F8 03   STA 03F8
D5B2   A9 29      LDA #$29
D5B4   8D FD 03   STA 03FD
D5B7   A0 00      LDY #$00
D5B9   B1 FA      LDA  (FA),Y
D5BB   A2 20      LDX #$20
D5BD   DD E7 D6   CMP D6E7,X
D5C0   F0 09      BEQ D5CB
D5C2   CA         DEX
D5C3   D0 F8      BNE D5BD
D5C5   4C 82 D6   JMP D682
D5C8   4C 54 D6   JMP D654
D5CB   BD 07 D7   LDA D707,X
D5CE   8D F5 03   STA 03F5
D5D1   BD 27 D7   LDA D727,X
D5D4   8D F6 03   STA 03F6
D5D7   BD 47 D7   LDA D747,X
D5DA   8D F7 03   STA 03F7
D5DD   A5 F4      LDA   F4
D5DF   85 EA      STA   EA
D5E1   A0 00      LDY #$00
D5E3   B1 FA      LDA  (FA),Y
D5E5   20 AB D6   JSR D6AB
D5E8   8C EA 03   STY 03EA
D5EB   8D EB 03   STA 03EB
D5EE   C6 EA      DEC   EA
D5F0   F0 1E      BEQ D610
D5F2   A0 01      LDY #$01
D5F4   B1 FA      LDA  (FA),Y
D5F4   B1 FA      LDA  (FA),Y
D5F6   20 AB D6   JSR D6AB
D5F9   8C ED 03   STY 03ED
D5FC   8D EE 03   STA 03EE
D5FF   C6 EA      DEC   EA
D601   F0 0D      BEQ D610
D603   A0 02      LDY #$02
D605   B1 FA      LDA  (FA),Y
D607   20 AB D6   JSR D6AB
D60A   8C F0 03   STY 03F0
D60D   8D F1 03   STA 03F1
D610   20 00 D0   JSR D000
```

```
D613  A2 00      LDX #$00
D615  BD E3 03   LDA 03E3,X
D618  20 09 D0   JSR D009
D61B  E8         INX
D61C  E0 1D      CPX #$1D
D61E  D0 F5      BNE D615
D620  A5 EB      LDA  EB
D622  30 15      BMI D639
D624  38         SEC
D625  E9 01      SBC #$01
D627  85 EB      STA  EB
D629  D0 0E      BNE D639
D62B  20 5A 1E   JSR 1E5A
D62E  C9 0D      CMP #$0D
D630  D0 03      BNE D635
D632  6C FC FF   JMP(FFFC)
D635  A9 10      LDA #$10
D637  85 EB      STA  EB
D639  20 63 1F   JSR 1F63
D63C  A5 FA      LDA  FA
D63E  CD F7 17   CMP 17F7
D641  D0 07      BNE D64A
D643  A5 FB      LDA  FB
D645  CD F8 17   CMP 17F8
D648  F0 07      BEQ D651
D64A  C6 F4      DEC  F4
D64C  D0 EB      BNE D639
D64E  4C 01 D5   JMP D501
D651  6C FC FF   JMP(FFFC)
D654  A5 FB      LDA  FB
D656  85 F9      STA  F9
D658  A0 01      LDY #$01
D65A  B1 FA      LDA  (FA),Y
D65C  10 02      BPL D660
D65E  C6 F9      DEC  F9
D660  18         CLC
D661  69 02      ADC #$02
D663  90 02      BCC D667
D665  E6 F9      INC  F9
D667  18         CLC
D668  65 FA      ADC  FA
D66A  90 02      BCC D66E
D66C  E6 F9      INC  F9
D66E  20 AB D6   JSR D6AB
D671  8D FC 03   STA 03FC
D674  8C FB 03   STY 03FB
D677  A5 F9      LDA  F9
D679  20 AB D6   JSR D6AB
D67C  8C F9 03   STY 03F9
D67F  4C 9B D5   JMP D59B
D682  A2 18      LDX #$18
D684  A0 00      LDY #$00
D686  B1 FA      LDA  (FA),Y
D688  3D AF D7   AND D7AF,X
D68B  5D C7 D7   EOR D7C7,X
D68E  F0 06      BEQ D696
D690  CA         DEX
D691  D0 F3      BNE D686
D693  4C 29 19   JMP 1929
D696  BD 67 D7   LDA D767,X
D699  8D F5 03   STA 03F5
D69C  BD 7F D7   LDA D77F,X
D69F  8D F6 03   STA 03F6
D6A2  BD 97 D7   LDA D797,X
D6A5  8D F7 03   STA 03F7
D6A8  4C DD D5   JMP D5DD
D6AB  AA         TAX
D6AC  4A         LSR  A
D6AD  4A         LSR  A
D6AE  4A         LSR  A
D6AF  4A         LSR  A
D6B0  20 B9 D6   JSR D6B9
D6B3  A8         TAY
D6B4  8A         TXA
D6B5  20 B9 D6   JSR D6B9
D6B8  60         RTS
D6B9  29 0F      AND #$0F
D6BB  C9 0A      CMP #$0A
D6BD  18         CLC
D6BE  30 02      BMI D6C2
D6C0  69 07      ADC #$07
D6C2  69 30      ADC #$30
D6C4  60         RTS
```

Tabellen würden dann einfach hexadezimal ausgedruckt.

Das zusätzliche Anzeigen des Maschinencodes pro Disassembler-Zeile geschieht im Programm von D5DD bis D60D. Abhängig davon, ob es sich um einen Ein-, Zwei- oder Dreibyte-Befehl handelt, wird der Instruktionscode in die Zellen 3EA/B, 3ED/E und 3F0/1 geladen.

Praktisch für das Arbeiten auf dem Bildschirm ist es, wenn nach jeweils einer vollgeschriebenen Seite das Programm abwartet, bis man das Ausgedruckte gelesen hat. Die Speicherzelle EB dient hierfür als ein Zähler, der auf 16 Zeilen gesetzt wird. Bei D620 wird er abgeprüft. Sollte ein negativer Wert gefunden werden, so ist der 16-Zeilen-Zyklus außer Betrieb. Wenn ein längeres Programm auf einem Drucker ausgegeben werden soll, so kann vor einem Start des Disassemblers EB mit „FF" geladen und der Disassembler bei D4F7 gestartet werden. Der Drucker arbeitet dann kontinuierlich bis zum Ende des Listings.

Sicher wird nicht jeder dieses Programm an derselben Speicherstelle verwenden wollen. Es ist jedoch sinnvollerweise so ausgelegt, daß es genau an einer Speicherseitengrenze endet (D7FF). Soll ein anderer Speicherbereich benutzt werden, so sind die höheren Adreßbytes zu ändern, überall, wo in der dritten Opcode-Zeile D4...D7 steht, und außerdem muß in Zelle D542 der Inhalt D5, welcher hier eine Speicherseite für Sprungziele bedeutet, für den neuen Platz umgeändert werden. Das Ausgabefeld im RAM-Bereich liegt bei 03E3 bis 03FF. Es sollte beibehalten werden, da sonst sehr viele Programmzeilen zu ändern wären. ■

```
Es folgen die Tabellen im Bereich D6C5...D7FF:
==============================================

D6C0  690769306OFFFFD7FFFFFFDF9F1F8D1F
D6D0  1F1C1C1C1C190514206CBE960A108011
D6E0  01040C141C1900000090B0F030D01050
D6F0  7018D858B8CA88E8C8EA480868284060
D700  AAA8BA8A9A9838F84242424242424242
D710  4243434343444449494E505050505252
D720  5454545454545353524343454D4E5056
D730  564C4C4C4C45454E4E4F48484C4C5454
D740  4141535858594545B4353514945 4C43
D750  5343444956585958595041504504953
D760  585958415341434445353524F4C4C4C
D770  4C434141414A5353524944434342 4A53
D780  4F54424F525344444444D534E444D5454
D790  4F4E4550504953455241434C41525958
D7A0  41504C4443505958524343585954524 9
D7B0  E3E3E3E3E3E3E3E3E3E3E3E3E3DFE7E7
D7C0  E3E7E7F3F3F7FFFF4181E1220142A0A2
D7D0  A1C10221614C848662E6C6E0C0242078
D7E0  0303030201020202020203020303010 2
D7F0  B7ADA4A478C86F7C8AB7B7A0A0A4B76F
```

Video-Ausgabe beim AIM-65

Der Mikrocomputer AIM-65 besitzt zwar ein 20stelliges alphanumerisches Display; beim Arbeiten mit dem Assembler oder in BASIC ist aber die Verwendung einer externen Video-Interface-Schaltung (z. B. VAB-2 von Mostek) recht praktisch. Leider besitzt das VAB-2 aber wiederum keine ASCII-Tastatur; es wäre daher nützlich, wenn das Video-Interface zusätzlich zum Display des AIM-65 mitliefe und dadurch die schon vorhandene AIM-Tastatur weiterbenützt werden könnte. Das Programm im *Bild* dient diesem Zweck. Bei der Inbetriebnahme ist wie folgt vorzugehen:

1. Das Programm im Bild laden (0170...01A4).

2. Um die Video-Ausgabe zu erreichen, schließt man das Interface an die serielle TTY-Schnittstelle des AIM an, schaltet es auf 300 Bd und startet das Programm an der Adresse 0170. Der TTY-Schalter auf der AIM-Platine bleibt dabei in Stellung KB.

3. Auf dem Bildschirm erscheint „Rockwell AIM-65". Alle Ein- und Ausgaben erfolgen jetzt parallel auf dem Video- und dem LED-Display.

4. Um die Ausgabe auf dem Bildschirm wieder zu unterbinden, startet man das Programm bei 0198. Das Video-Interface ist dann abgeschaltet; auf dem AIM-Display erscheint „Rockwell AIM-65".

Mit diesem Programm ist es möglich, ein Video-Interface oder einen Drucker mit serieller Schnittstelle zusammen mit der AIM-65-Tastatur zu betreiben

```
0170 A9 LDA #87
0172 8D STA A406
0175 A9 LDA #01
0177 8D STA A407
017A A9 LDA #0C
017C 8D STA A417
017F A9 LDA #C2
0181 8D STA A418
0184 6C JMP (FFFC)
0187 20 JSR EEA8
018A 29 AND #7F
018C C9 CMP #0D
018E D0 BNE 0197
0190 48 PHA
0191 A9 LDA #0A
0193 20 JSR EEA8
0196 68 PLA
0197 60 RTS
0198 A9 LDA #05
019A 8D STA A406
019D A9 LDA #EF
019F 8D STA A407
01A2 6C JMP (FFFC)
```

Selbstverständlich eignet sich die Routine auch für das Ausdrucken von Programmen auf einem Drucker mit serieller Schnittstelle. Die Baudrate kann an den Adressen 017B und 0180 nach den Angaben im AIM-Handbuch, auf den Seiten 9...29, geändert werden. Fe.

„Pseudo-Reset" beim PET-2001

Wenn Sie PET-Benutzer sind und gerne in Maschinensprache programmieren, werden Sie sich sicherlich schon darüber geärgert haben, daß die Commodore-Computer keine Reset-Taste eingebaut haben. Wenn die CPU durch einen kleinen Programmierfehler (falscher Sprung o. ä.) irgendwo „rotiert" und nur noch durch einen nicht-maskierbaren Interrupt per Ein-Aus-Schalter auf den richtigen Weg gebracht werden kann, dann haben sicher auch Sie wegen des gelöschten Programms die Lust verloren.

Natürlich kann man sich eine Reset-Taste einbauen (der dafür notwendige Anschluß liegt an A27 der Steckleiste für die Speichererweiterung), aber wer möchte seinen PET schon anbohren oder gar viel Geld für den notwendigen Stecker ausgeben? Außerdem löscht der Power-on-reset den Basic-Arbeitsspeicher, so daß dort eventuell noch vorhandene Programme oder Variablen verloren sind. Wenn man sich mit einigen Einschränkungen abfindet, läßt sich eine Software-Lösung verwenden.

Diese Einschränkungen betreffen den Interrupt-Vektor. Wenn der Computer „Harakiri" läuft und dabei den Interrupt-Vektor in 0219 und 021A überschreibt, so ist das folgende Programm nicht zu gebrauchen. Auch wenn durch das „Totprogramm" der Stackpointer und/oder der Stack getötet werden, hilft nur noch der bekannte Griff zur Rückseite des PET. Wenn jedoch wie in vielen Fällen der Tod auf eine endlose Schleife, hervorgerufen vielleicht durch eine falsche Bedingungsstellung, folgt, dann findet, obwohl der Rechner keine Befehle mehr annimmt, trotzdem ein Interrupt mit anschließender Tastaturabfrage statt. Das nutzt das folgende Programm aus. Es wird im zweiten Cassettenspeicher abgelegt und mit SYS826 gestartet.

Zunächst wird der Interrupt-Vektor auf den zweiten Cassettenpuffer gerichtet. Dies geschieht über den EOR-Befehl, so daß das Programm mit nochmaligem SYS826 wieder desaktiviert werden kann. Dies ist notwendig, wenn man mit dem Cassettenrecorder arbeiten will. Bei LOAD, SAVE, VERIFY und OPEN X, 1, Y wird der Interrupt-Vektor benötigt. Zur Kontrolle, ob das Programm gerade aktiviert ist oder nicht, wird bei jedem Durchlauf das rechte obere Feld des Bildschirms invertiert, so daß es schnell flackert. So kann man bei einem „Harakiri"-Programm gleich feststellen, ob der Pseudo-Reset noch möglich ist. (Wenn das Feld flackert, dann ja, sonst .... siehe oben!)

Falls der Programmzähler also durch mildernde Umstände das neue Interruptprogramm noch erreicht, erfolgt eine Abfrage der zuletzt gedrückten Taste. Die Pseudo-Reset-Taste ist Shift-Return, es müssen also Shift und Return gleichzeitig gedrückt werden, um dem Programmverlust zu entgehen. Dann wird schlicht und ergreifend die Interrupt-Rücksprungadresse im Stack geändert. Der Rechner kommt nach dem RTI nicht mehr in das Totprogramm zurück. Er springt stattdessen in die Basic-Warm-Start-Routine und kehrt mit „Ready" auf den Bildschirm zurück. Damit er dann nicht wieder in diese Routine springt, weil Shift/Return sicher immer noch gedrückt ist, wird noch eine Tastaturabfrage eingesetzt, die dafür sorgt, daß die Warm-Start-Routine erst aufgesucht wird, wenn Shift und Return wieder losgelassen worden sind. Diese zweite Tastaturabfrage sieht ganz anders aus als die erste, auch das hat seinen Grund. Während der Ausführung eines Interrupt-Programms ist die normale Abfrage der Tastatur außer Betrieb (sie wird erst mit JMP E685 wieder gestartet). Daher erfolgt die zweite Abfrage direkt über das PIAL- und das PIAK-Register im 6520.

Nach Eingabe und Start des Programms kann eine Funktionskontrolle mit folgender Befehlsfolge stattfinden:
NEW:POKE1026,255:LIST

Der Rechner wird wirres Zeug ausdrucken, weil der List-Pointer mit dem o.a. Befehl auf die Pfade der Unendlichkeit geschickt wird. Nach Druck auf Shift-Return und anschließendem Loslassen meldet sich der PET mit „Ready" wieder.

Das folgende Basic-Programm erzeugt auf dem Bildschirm ein optisch sehr reizvolles Muster, auch daraus kann der PET mit dem Pseudo-Reset wieder herausgeangelt werden:

```
10 POKE 32768,76:POKE32769,0:
POKE32770,128
20 SYS32768
RUN
```

Und nun viel Spaß beim Experimentieren!

Stefan Burmeister

Pseudo — Reset

```
C*   PC  SR AC XR YR SP
.;  C6ED 50 11 53 45 FE
.
033A:   78          SEI             : VERHINDERE INTERRUPT
033B:   AD 19 02    LDA  $0219      : AENDERE INTERRUPT-VEKTOR
033E:   49 C8       EOR  #$C8       :
0340:   8D 19 02    STA  $0219      :
0343:   AD 1A 02    LDA  $021A      : NEUER INTERRUPT AUF $034D
0346:   49 E5       EOR  #$E5       :
0348:   8D 1A 02    STA  $021A      :
034B:   58          CLI             : LASSE NEUEN INTERRUPT ZU
034C:   60          RTS             : RUECKKEHR ZU BASIC
034D:   AD 27 80    LDA  $8027      : INVERTIERE RECHTES,
0350:   49 80       EOR  #$80       : OBERES BILDSCHIRMFELD
0352:   8D 27 80    STA  $8027      :
0355:   AD 04 02    LDA  $0204      : ABFRAGE, OB SHIFT GEDRUECKT
0358:   F0 07       BEQ  $0361      :
035A:   AD 03 02    LDA  $0203      : ABFRAGE, OB RETURN GEDRUECKT
035D:   C9 1B       CMP  #$1B       :
035F:   F0 03       BEQ  $0364      :
0361:   4C 85 E6    JMP  $E685      : WENN NICHT, DANN IRQ-RUECKKEHR
0364:   BA          TSX             : LADE INDEX X MIT STACKPOINTER
0365:   A9 C3       LDA  #$C3       : ADRESSE DES BASIC-WARM-STARTS
0367:   9D 06 01    STA  $0106,X    : ANSTELLE DES ALTEN IRQ-RETURNS
036A:   A9 8B       LDA  #$8B       :
036C:   9D 05 01    STA  $0105,X    :
036F:   A9 08       LDA  #$08       : EIGENE TASTATURABFRAGE
0371:   8D 10 E8    STA  $E810      : (NORMALE ABFRAGE IM
0374:   AD 12 E8    LDA  $E812      : INTERRUPTPROGRAMM NICHT
0377:   C9 DF       CMP  #$DF       : MOEGLICH)
0379:   F0 F4       BEQ  $036F      : SHIFT/RETURN NOCH GEDRUECKT ?
037B:   4C 85 E6    JMP  $E685      : NEIN! DANN IRQ-RUECKKEHR
B*   PC  SR AC XR YR SP             :
.;  07B8 32 35 00 00 FE
.
.
READY.
```

Maschinenprogramm für die zyklische Abfrage von Shift-Return während der Timer-Interrupt-Routine des PET. Bei 036B muß es statt 8B richtig 8A heißen

Das KIM-1-Tastenfeld ist nach einer gewissen Benutzungsdauer nicht mehr ganz prellfrei. Im folgenden wird daher eine Methode beschrieben, wie man mit einer gewöhnlichen ASCII-Tastatur alle Funktionen des Monitorprogramms nutzen kann.

Ing. grad. H. W. Kruber

ASCII-Tastatur steuert KIM-1

Die handelsüblichen ASCII-Tastaturen haben meist eine Parellel-Schnittstelle. Der Ausgang hat 7 bit und positiven oder negativen Strobeanschluß.

Bekanntlich enthält das KIM-Monitor-Programm nur einen seriellen Eingang für ASCII-Zeichen. Der Anschluß der Tastatur muß deshalb per Software über den I/O-Port A erfolgen. Der notwendige Hardware-Aufwand ist gering: Er besteht aus einer Steckleiste, die mit dem Applikations- und Expansions-Anschluß des KIM verdrahtet werden muß. Dieser Stecker wird auf den Steckkamm der Tastatur aufgesteckt (Bild 1).

Das ausgearbeitete Programm (Bild 2) besteht aus zwei Interruptroutinen. Der Interrupt wird durch Aktivierung des NMI-Anschlusses (Kontakt 6 des Expansions-Anschluß am KIM) ausgelöst. Die Aktivierung besorgt der negative Strobe, den die Tastatur abgibt.

Die zwei Interruptroutinen haben unterschiedliche Aufgaben. Die erste dient der Eingabe von Adressen und Daten sowie der Erkennung der übrigen simulierten Tasten des KIM-Keyboards. Sie erkennt, welche Tasten gedrückt sind und wandelt ASCII-Zeichen in Hex-Zeichen. Auch erkennt sie, ob beabsichtigt wird, ASCII-Zeichen als String zu schreiben. Die Information hierfür wird durch Drücken folgender Tasten der Tastatur gegeben:

CTRL-Taste + A = Adresseneingabe
CTRL-Taste + D = Dateneingabe
CTRL-Taste + G = Go
CTRL-Taste + S = Step
CTRL-Taste + P = PC
@ = Es wird ein ASCII-String eingegeben.

Es werden also alle Tasten des KIM-Keyboard simuliert, außer der Stop- und Reset-Taste. Der SST-Schalter wird auch weiter benutzt. Bei diesen Tasten und dem SST-Schalter stören Prellungen nicht. Es ist beabsichtigt, diese Taste später, wenn ein Gehäuse angeschafft wird, ebenfalls zu ersetzen. Auf der Tastatur sind zwei Tasten für individuelle Benutzung frei und eignen sich hierfür bestens. Ein Schalter in der Frontplatte eines Gehäuses kann dann den SST-Schalter ebenfalls ersetzen.

Die Routine 2 ist, wie bereits erwähnt, für die Eingabe eines ASCII-Zeichen String vorgesehen. Der String kann an jeder beliebigen Stelle des RAM-Speicherbereichs aufgebaut werden. Die Anfangsadresse wird in den Zero-Page Adressen 00EB und 00EC hinterlegt. Wie beim KIM gewohnt, kommt das niedere Byte in 00EB und das höherwertige Byte in 00EC. Ein Zeichenzähler kommt in die Adresse 00EA. Er ist stets vor Beginn einer ASCII-Eingabe auf 00 zu setzen.

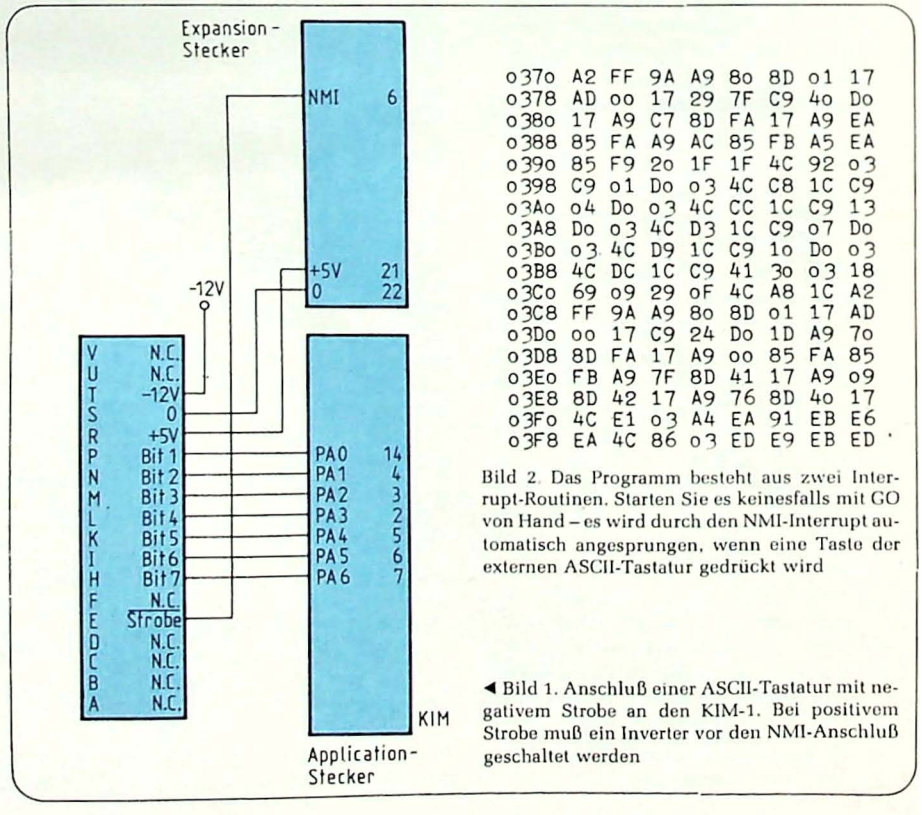
Wird die @-Taste gedrückt, so erscheint im Display „AC EA xx". Das bedeutet: es kann nun mit ASCII-Zeichen geschrieben werden, der Zeichen-Zähler steht auf dem Wert xx. Hat man vorher vergessen, diesen auf 00 zu stellen, wird man nun daran erinnert.

Ist die Eingabe der ASCII-Zeichen beendet und man will weitere Daten eingeben, wird in Verbindung mit der Shift-Taste die Taste $ gedrückt. Es erscheint im Display der Buchstabe H. Drückt man nun die Adreß-Taste, erscheint im Display die Adresse 0000 mit ihrem Inhalt. Es wird nun wie früher gewohnt eingegeben.

Ein Tip für eine bequeme Eingabe: Geben Sie zunächst nur die Adressen 0370 bis 03C6 ein. Setzen sie den NMI-Vektor 17FA auf 70 und die Adresse 17FB auf 03. Nun können Sie den Rest des Programmes bereits mit der Tastatur eingeben.

Wer das Programm nicht zwischen den Adressen 0370 und 03FB unterbringen will, muß es abändern. Geändert werden müssen die Adressen: 0382, 0396, 0397, 03D7, 03F1, 03F2, 03FA, 03FB, 17FA und 17FB.

Nach der Umschaltung von ASCII-Zeichen auf Eingabe darf nicht die GO-Taste gedrückt werden. Ist dies versehentlich doch geschehen, muß erst die Reset-Taste am KIM-Keyboard gedrückt werden. Es erscheint dann wieder die Anzeige der Adresse 0000 und ihr Inhalt.



◄ Bild 1. Anschluß einer ASCII-Tastatur mit negativem Strobe an den KIM-1. Bei positivem Strobe muß ein Inverter vor den NMI-Anschluß geschaltet werden

```
037o A2 FF 9A A9 8o 8D o1 17
0378 AD oo 17 29 7F C9 4o Do
038o 17 A9 C7 8D FA 17 A9 EA
0388 85 FA A9 AC 85 FB A5 EA
039o 85 F9 2o 1F 1F 4C 92 o3
0398 C9 o1 Do o3 4C C8 1C C9
03Ao o4 Do o3 4C CC 1C C9 13
03A8 Do o3 4C D3 1C C9 o7 Do
03Bo o3 4C D9 1C C9 1o Do o3
03B8 4C DC 1C C9 41 3o o3 18
03Co 69 o9 29 oF 4C A8 1C A2
03C8 FF 9A A9 8o 8D o1 17 AD
03Do oo 17 C9 24 Do 1D A9 7o
03D8 8D FA 17 A9 oo 85 FA 85
03Eo FB A9 7F 8D 41 17 A9 o9
03E8 8D 42 17 A9 76 8D 4o 17
03Fo 4C E1 o3 A4 EA 91 EB E6
03F8 EA 4C 86 o3 ED E9 EB ED
```

Bild 2. Das Programm besteht aus zwei Interrupt-Routinen. Starten Sie es keinesfalls mit GO von Hand – es wird durch den NMI-Interrupt automatisch angesprungen, wenn eine Taste der externen ASCII-Tastatur gedrückt wird

Autostart-Einrichtung für AIM-65 und PC-100

Mit geringstem Aufwand ist es möglich, mit den Mikrocomputern AIM-65 und PC-100 Programme von der Kassette so zu laden, daß diese sich selbst starten. Auch bereits vorhandene Programme können durch einfaches „Vorspannen" eines Datenblocks dafür erweitert werden, so daß man nicht erst immer die Startadresse einstellen und „G" drücken muß. Diese Aufgabe übernimmt ein kleines Steuerprogramm.

Der Trick beim Selbststart ist, daß die Kassette den DILINK-Vektor zunächst auf die Startadresse des Steuerprogramms umsetzt. Da man diesen Vektor nicht von Hand ändern kann, weil sich das System sonst „aufhängt", ist in *Bild 1* ein Programm angegeben, das nichts anderes tut, als das in *Bild 2* wiedergegebene Steuerprogramm nebst dem geänderten DILINK-Vektor (A406/A407) auf Kassette abzuspeichern. Das Steuerprogramm in Bild 2 startet sich beim Laden dann selbst, lädt das nachfolgende Programm (das den gleichen Filenamen haben muß) und springt zu dessen Startadresse, hier 0000.

Will man bei einem vorhandenen, bereits irgendwo auf Kassette befindlichen Programm die Autostart-Einrichtung ergänzen, so ist wie folgt vorzugehen:

1. Die Daten von 0F00 bis 0F4F laut Bild eingeben, dann an der Adresse 0F00 mit „G" starten.

2. Nach „F=" den Filenamen des zu startenden Programms eingeben und die Kassette etwa zwei Sekunden vor dieses positionieren.

3. Recorder auf Aufnahme schalten und nach „T=" wie gewohnt z.B. zweimal Return drücken.

4. Sobald der Monitor-Prompt wieder im Display erscheint, Kassettenrecorder sofort anhalten, damit das nachfolgende Programm nicht überschrieben wird.

Die Startadresse des zu ladenden Programms findet sich in Bild 1 an den Adressen 0F38 (L) und 0F39 (H). Wer Treiber- und Steuerprogramm in einen anderen Adressenbereich legen will, z.B. weil das zu ladende Programm ebenfalls bei 0F00 liegt, muß selbstverständlich auch den Tape-Buffer-Block (0F1E...0F4F) entsprechend ändern – hier sei empfohlen, sich einmal das Kapitel „Kassetten-Objektcode-Format" im AIM-65-Handbuch anzusehen.

Sollten Ihre Programme gewöhnlich mit ganz unterschiedlichen Adressen gestartet werden müssen, so kann es recht zeitraubend sein, jedesmal die neue Startadresse bei 0F39 einzutragen und auch die Prüfsumme bei 0F3B entsprechend zu korrigieren. Das zusätzliche Programm in *Bild 3* (Startadresse 0F70) nimmt diese Mühe ab; es erwartet vom Benutzer zunächst die Eingabe einer (stets vierstelligen) Startadresse des nachfolgenden Programms, das automatisch gestartet werden soll, errechnet dann die passende Prüfsumme und springt dann nach 0F00, um das schon beschriebene Programm zur Generierung des Autostart-Datenblocks abzuarbeiten. Die Bedienung vereinfacht sich dadurch wesentlich. Selbstverständlich ist es auch möglich, daß der Autostart-Generator sich bei 0F70 selbst startet – dazu muß man ihm lediglich den von ihm selbst erzeugten Datenblock „vorspannen".

Herwig Feichtinger

```
<K>•=F00
/12
 0F00 A9 LDA #54
 0F02 8D STA A413
 0F05 A2 LDX #01
 0F07 20 JSR E8A2
 0F0A 20 JSR E56F
 0F0D BD LDA 0F19,X
 0F10 20 JSR F18B
 0F13 E8 INX
 0F14 E0 CPX #36
 0F16 D0 BNE 0F0D
 0F18 20 JSR E50A
 0F1B 20 JSR E182

<M>=0F1E 0D 3B 18 0F
< > 0F22 50 A9 05 8D
< > 0F26 06 A4 A9 EF
< > 0F2A 8D 07 A4 A9
< > 0F2E 54 8D 12 A4
< > 0F32 20 2F E3 20
< > 0F36 E9 E2 4C 00
< > 0F3A 00 0A D5 0D
< > 0F3E 3B 02 A4 06
< > 0F42 50 0F 01 0B
< > 0F46 0D 3B 00 00
< > 0F4A 03 00 03 0D
< > 0F4E 00 00
```

◄ Bild 1. Treiberprogramm zum Erzeugen des in Bild 2 wiedergegebenen Steuerprogramms. Es generiert auch einen geänderten DILINK-Vektor auf Kassette, so daß sich das Steuerprogramm später selbst starten kann

```
<K>•=0F50
/09
 0F50 A9 LDA #05
 0F52 8D STA A406
 0F55 A9 LDA #EF
 0F57 8D STA A407
 0F5A A9 LDA #54
 0F5C 8D STA A412
 0F5F 20 JSR E32F
 0F62 20 JSR E2E9
 0F65 4C JMP 0000
```

Bild 2. Dieses Programm braucht man nicht einzugeben! Es wird vom Treiberprogramm in Bild 1 automatisch auf Kassette generiert, lädt das nachfolgende Programm und startet es – hier an der Adresse 0000

```
<M>=0F70 A9 00 8D 3B
< > 0F74 0F 8D 3C 0F
< > 0F78 20 F0 E9 A2
< > 0F7C 00 BD B0 0F
< > 0F80 F0 06 20 7A
< > 0F84 E9 E8 D0 F5
< > 0F88 20 FD E3 8D
< > 0F8C 3A 0F 20 FD
< > 0F90 E3 8D 39 0F
< > 0F94 A2 1B 18 AD
< > 0F98 3C 0F 7D 20
< > 0F9C 0F 8D 3C 0F
< > 0FA0 AD 3B 0F 69
< > 0FA4 00 8D 3B 0F
< > 0FA8 CA 10 EB 4C
< > 0FAC 00 0F 00 00
< > 0FB0 53 54 41 52
< > 0FB4 54 3D 00 00
<•>=F70

<G>/

START=0200 F=RTTY T=1
```

Bild 3. Zusatzprogrammteil für erhöhten Komfort: Nach dem Start bei 0F70 kann eine beliebige Adresse eingegeben werden, an der das auf der Kassette nachfolgende Programm gestartet werden soll

Die große Stärke der CPU 6502 ist das breite Spektrum von Adressierungsarten, die unter exzessiver Benutzung der „Page Zero" verwirklicht wurden. Bisweilen ist der Zugang zur Page Zero jedoch verbaut, etwa dann, wenn mehrere Teilprogramme zusammenarbeiten sollen und aus ihrer Dokumentation nicht hervorgeht, welches Programm nun welche Speicherstellen der Nullseite für sich in Anspruch nimmt. Das führt zu Schwierigkeiten. Die allerdings sind lösbar – und wie, das zeigt der folgende Beitrag.

Hans Georg Joepgen

6502 simuliert neue Adressierungsart

Im Herbst 1979 erschien die Beschreibung einer eleganten Ausgaberoutine für 6502-Maschinen [1], die sich gegenüber den bis dahin in der Öffentlichkeit bekanntgewordenen Ausgabeverfahren dadurch auszeichnete, daß die zu druckenden Zeichen im aufrufenden Hauptprogramm abgelegt werden und das ausgebende Unterprogramm selbsttätig die Programmkontrolle an jene Speicherstelle zurückgibt, die dem Textstring folgt; wobei der Programmierer, und hier liegt der Hauptvorzug dieses von seinem Autor „PRTSTR" genannten Programmsegments, völlig freie Hand hat, was die Länge der auszugebenden Zeichenkette angeht. Der Rechner interpretiert die dem Unterprogramm-Aufruf folgenden Speicherinhalte so lange als ASCII-Codes, bis er auf eine Endmarkierung stößt. PRTSTR hat als Standard-Ausgabeprozedur unterdessen in vielen Entwicklungslabors und Hobby-Computer-Clubs Einzug gehalten. Rolf-Fr. Matthaei veröffentlichte eine Z-80-Version für den TRS-80, die das HL-Doppelregister als Zwischenspeicher benutzt und mit weniger Anweisungen auskommt als die 6502-Version.

Für seine trickreiche Technik der „freischwebenden Rückkehr" ins aufrufende Hauptprogramm bedient sich PRTSTR freilich der Page Zero, und das führte beim Verfasser dieses Beitrags gelegentlich zu Ärger, weil Disk-Operating-Systeme und neue Hochsprachen-Interpreter justament immer jene Speicherstellen für sich beanspruchten, die als Plätze für PRTSTR benötigt wurden.

Als auch nach mehrmaliger Verlegung immer wieder Kollisionen auftraten, wenn neuerworbene Software mit dieser Ausgabe-Routine zusammenarbeiten sollte, da entschied sich der Autor, eine Neufassung von PRTSTR zu schreiben, die völlig ohne die vertrackte Page Zero auskommen sollte.

Den Ablaufplan zu „Superprint" zeigt die *Tabelle*. Als kritisch erwies sich die Implementierung der Teilaufgabe Nummer 4 aus dem Ablaufplan: „Lade Akkumulator mit dem Inhalt jener Speicherstelle, deren Adresse du in CODE 1 und CODE 2 findest" – mit anderen Worten, hier war indirekte Adressierung bei einem Ladevorgang verlangt. Nun kennt der 6502 zwar mancherlei indirekte Adressierungsarten bei Ladevorgängen unter Mitwirkung des x- oder des y-Registers, die man gegebenenfalls durch Nullsetzen

Tabelle: Ablaufplan zu „Superprint"

SPRINT (Beginn von 'Superprint')	1. Hole niederwertiges Byte der Rückkehradresse und kopiere es in eine Speicherstelle CODE 1 2. Hole höherwertiges Byte der Rückkehradresse und kopiere es in eine Speicherstelle CODE 2. Datenquelle bei beiden Operationen ist der Stapelspeicher (STACK). Es wird die Operation PLA benutzt, die den Stapelzeiger (Stack Pointer) selbsttätig korrigiert.
SCHLEI (Schleifenanfang)	3. Erhöhe das Doppelbyte in den Speicherstellen CODE 1 (niederwertiges Byte) und CODE 2 (höherwertiges Byte) um Eins. Bediene dich dabei einer Subroutine SUBRT. 4. Lade Akkumulator mit dem Inhalt jener Speicherstelle, deren Adresse du in CODE 1 und CODE 2 findest. 5. Befindet sich nun das vereinbarte Endezeichen ENDZ im Akkumulator (bei der Realisierung des Programms wurde die Anweisung NO OPERATION, NOP, mit dem Hex-Maschinencode EA benutzt)? 6. Wenn n e i n: Interpretiere Akkuinhalt als ASCII-Code und veranlasse mit Hilfe der maschinenspezifischen Ausgaberoutine den Ausdruck des entsprechenden Zeichens. Springe zurück zum Schleifenanfang SCHLEI. 7. Wenn j a: Erhöhe den Wert des Doppelbytes in CODE 1, CODE 2 um 1 (Subroutine SUBRT). 8. Springe zurück ins Hauptprogramm, und zwar unmittelbar an die Stelle nach dem Schlußzeichen ENDZ. Die Adresse dieser Stelle findest du in CODE 1 (niederwertiges Byte) und CODE 2 (höherwertiges Byte).
SUBRT	9. Erhöhe Inhalt von CODE 1 um Eins. War das Ergebnis Null, dann erhöhe auch Inhalt von CODE 2 um Eins. Springe zurück nach jenem SPRINT-Segment, von dem der Aufruf kam.

neutralisieren kann; alle diese Operationen verlangen aber die Mitwirkung von mindestens einer Page-Zero-Speicherstelle.

Der Ausweg, der gefunden wurde, geht von einer Doppelinterpretation des LDA-Befehls aus (LDA – absolut: Lade Akkumulator mit dem Inhalt jener Speicherstelle, deren Adresse in jenen Speicherstellen steht, die dem Befehl folgen; das zweite Byte der Dreiergruppe ist der niederwertige Teil der Adresse, das letzte der Gruppe ist der höherwertige Adreßteil).

Wie *Bild 1* zeigt, werden die drei Speicherstellen CODE/CODE 1/CODE 2 einmal als klassischer LDA-Befehl aufgefaßt. Wenn sich der Rechner den Opcode mit dem Hexwert AD aus dem Speicher holt, erkennt er aufgrund seiner internen Befehlsdekodierung, daß die Gruppe als Drei-Byte-Befehl zu behandeln ist, und so verfährt er auch. Wenn nun aber von einer anderen Stelle des Programms aus die Speicherstellen CODE 1 und CODE 2 als schlichter Zweibyte-Zähler, als Zeiger gewissermaßen, behandelt werden und damit als klassische Datenspeicherstelle, in der man die Adresse des Ladeoperation-Operanden ablegt, dann müßte es doch eigentlich möglich sein, den Rechner hier ein wenig übers Ohr zu hauen: Der CPU eine Adressierungsart abzuverlangen, über die sie von Haus aus nicht verfügt. Das „Lade indirekt" nämlich, ohne Wenn und Aber, ohne x, y und Zero.

Bild 2 zeigt die neue Routine Superprint, geschrieben mit Hilfe des Six-Label-Assemblers der Firma Microproducts auf einer Maschine ITT 2020 Integer. *Bild 3* bringt eine kleine Prüfroutine, die dem Test von SPRINT dient.

Die Subroutine SPRINT, hier für die Bedürfnisse der ITT-2020-Integer und des Apple-II-Standard ausgeführt, kann problemlos für andere 6502 Maschinen wie KIM, AIM-65 oder Siemens PC-100 geändert werden, indem man – gegebenenfalls nach Verschieben in einen angenehmeren Speicherbereich – die Labeldefinition in Zeile 130 so ändert, daß sie als DRUCK die Adresse der betreffenden Druckroutine aus dem Betriebsprogramm des jeweiligen Mikrocomputers nennt. Bei Assemblierung von Hand ist das zweite und das dritte Byte des Maschinencodes in Zeile 260 (Adressen 0814 und 0814) dergestalt zu ändern, daß man zuerst den niederwertigen und dann den höherwertigen Teil der Adresse der Ausgaberoutine im Monitor einsetzt, für den KIM zum Beispiel bei Teletype-Betrieb A01E.

Was die Namensgebung für das angewandte Adressierungsverfahren angeht, so dankt der Verfasser Herrn Diplominformatiker Peter Müller in Stuttgart für den Vorschlag, von „Absolut-modifizierter Adressierung" (Absolute-modified) zu sprechen und für die entsprechende Ladeoperation beispielsweise das Kürzel LDA, M in Assembler-Niederschriften zu verwenden.

Literatur

[1] *Franke, Erich H.:* PRTSTR – Einfache Textausgabe in Assemblerprogrammen in „Hobbycomputer 2", Sonderheft der FUNKSCHAU, Franzis-Verlag, München 1979.

[2] *MOS Technology:* „Programmier-Handbuch", 1. Deutsche Auflage 1977, Vertrieb: MCDS Mikrocomputer Systeme Darmstadt.

[3] *Siemens:* Bedienungsanleitung Personal Computer PC 100, Vorläufige Ausgabe 1979/1980, München.

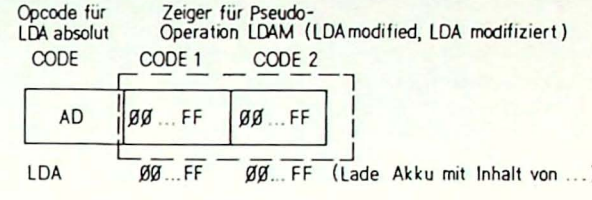
[4] *Apple Computer Incorporated:* Apple II Reference Manual, Januar 1978, Cupertino (Kalifornien).

[5] *Dougherty, William E:* The Apple II Monitor Peeled, Herausgeber und Vertrieb: Verfasser, 14349 San José Street, Mission Hills, California 91345, USA.

[6] *Reynolds, William:* Disassembling the DOS 3.2. „Micro, the Magazine of the Apple, KIM, PET", Heft 17, Band 1979, Micro Publishing, Chelmsford, Massachusetts, USA. Europa-Vertrieb Micro Shop Bodensee, D 7778 Markdorf.

[7] *Matthaei, Rolf-Fr.:* Textausgabe in Maschinensprache. FUNKSCHAU 1980, Heft 1.

Bild 1. Absolut-modifizierte Adressierung: Ein Programmtrick, mit der sich die CPU 6502 zu einer Zugriffsart bewegen läßt, über die sie von Haus aus nicht verfügt




```
                        0010  ; S P R I N T
                        0020  ; ===========
                        0030  ;
                        0040  ; "SUPERPRINT",
                        0050  ; DIENSTROUTINE
                        0060  ; ZUR AUSGABE VON
                        0070  ; TEXT UND TAFELN. -
                        0080  ;
                        0090  ; V2.01, 12.1 1980
                        0100  ; H.-G. JOEPGEN. -
                        0110  ;
                        0120  ;-------------------
                        0130  DRUCK  EQU FDED
                        0140  ENDZ   EQU EA
                        0150  ;-------------------
0800  68          0160  SPRINT PLA
0801  8D0C08      0170         STA CODE1
0804  68          0180         PLA
0805  8D0D08      0190         STA CODE2
0808  201E08      0200  SCHLFE JSR SUBRT
080B  AD          0210  CODE   DFD AD
080C  00          0220  CODE1  DFD 00
080D  00          0230  CODE2  DFD 00
080E  C9EA        0240         CMP #ENDZ
0810  F006        0250         BEQ ENDE
0812  20EDFD      0260         JSR DRUCK
0815  4C0808      0270         JMP SCHLFE
                  0280  ;-------------------
0818  201E08      0290  ENDE   JSR SUBRT
081B  6C0C08      0300         JMP (CODE1)
                  0310  ;-------------------
081E  EE0C08      0320  SUBRT  INC CODE1
0821  D003        0330         BNE SUBEND
0823  EE0D08      0340         INC CODE2
0826  60          0350  SUBEND RTS
                  0360  ;-------------------

LABEL  TABLE
DRUCK  FDED       CODE1  080C
ENDZ   00EA       CODE2  080D
SPRINT 0800       ENDE   0818
SCHLFE 0808       SUBRT  081E
CODE   080B       SUBEND 0826
```

◄ Bild 2. SUPERPRINT in Assembler-Sprache: Hier für den ITT-2020. Allein durch Änderung zweier Speicherstellen läßt sich das Programm auch anderen 6502-Mikrocomputern anpassen

```
0010  ; T E S T
0020  ; =======
0030  ;
0040  ; ERPROBUNG VON
0050  ; "SPRINT"
0060  ;
0070        ORG 900
0080  SPRINT EQU 800
0090        JSR SPRINT
0100        DFD " EINS "
0110        NOP
0120        JSR SPRINT
0130        DFD " ZWEI "
0140        NOP
0150        JSR SPRINT
0160        DFD 8D
0170        DFD "ENDE. -"
0180        NOP
0190        RTS


*900G
 EINS  ZWEI
ENDE. -
```

Bild 3. Ein Probelauf ergibt: SUPERPRINT benimmt sich erwartungsgemäß

In den FUNKSCHAU-Heften 15 und 16/1979 fanden sich ein Empfangs- und ein Sendeprogramm sowie eine einfache Interface-Schaltung mit PLL-Decoder, um die Mikrocomputer AIM-65 und PC-100 als Funkfernschreib-Terminal verwenden zu können. Das im folgenden beschriebene Programm gestattet nun auch die gleichzeitige Ausgabe von gesendeten und empfangenen Texten auf Display und Thermodrucker.

Dr. Martin Costabel

RTTY mit dem AIM-65 – jetzt mit Drucker

Die Ursache, warum das in Heft 15/1979 der FUNKSCHAU veröffentlichte Funkfernschreib-Empfangsprogramm als Ausgabemedium nur das Display, nicht aber den Drucker verwenden konnte, ist darin zu suchen, daß die Zeichenausgabe zwischen zwei empfangenen Zeichen bzw. während eines Stop-Bits erfolgen muß. Diese Bedingung läßt sich bei der doch recht langsamen Druckerausgabe nicht einhalten: Der AIM-65-Drucker arbeitet grundsätzlich zeilenweise, und die Ausgabe einer Zeile dauert viel länger als ein Baudot-Zeichen bei 45 Bd oder 50 Bd. Ein unlösbares Problem? Keineswegs! Schließlich verfügt der AIM-65 über mehrere programmierbare Timer (VIA 6522), die auch Interrupts auslösen können.

Das Programm in *Bild 1* erfordert einen Empfangskonverter (z. B. aus FUNKSCHAU 1979, Heft 16, „AIM-Erweiterungsplatine"), der an PA 0 und CA 2 (Pins 14 und 21 des Application Connector) angeschlossen wird. Die Verbindung nach CA 2 sorgt hier dafür, daß das Startbit des gerade empfangenen Zeichens einen Interrupt auslösen kann, so daß ein Sprung an die Interrupt-Routine an der Adresse 0400 erfolgt. Sobald das Startbit erkannt ist, wird mehrmals der Interrupt-Timer gestartet, um alle fünf Datenbits des Baudot-Zeichens einzulesen. Auch die Umwandlung in den ASCII-Code ge-

```
0000 A9 57 85 E4 | 0400 EA 48 8A 48 | 0200 A9 A1 8D 04 | 0280 C5 FD F0 0F
0004 A9 00 8D 0B | 0404 98 48 A5 E1 | 0204 A4 A9 02 8D | 0284 85 FD C9 00
0008 A0 8D 0C A0 | 0408 C9 07 30 0E | 0208 05 A4 A9 00 | 0288 F0 04 A9 1B
000C 85 FD 85 E2 | 040C C6 E1 A9 01 | 020C 85 E8 85 E9 | 028C D0 02 A9 1F
0010 85 E3 8D 04 | 0410 8D 0E A0 A5 | 0210 20 EF EC A9 | 0290 20 67 02 68
0014 A4 A9 04 8D | 0414 E4 4A 65 E4 | 0214 80 8D 02 A0 | 0294 C9 08 D0 05
0018 05 A4 A9 A1 | 0418 D0 4A AD 01 | 0218 A9 C0 8D 0B | 0298 20 67 02 A9
001C 8D 0E A0 A6 | 041C A0 29 01 18 | 021C A0 8D 0E A0 | 029C 02 20 67 02
0020 E2 E4 E3 F0 | 0420 F0 01 38 66 | 0220 A9 00 8D 05 | 02A0 60 2C 09 A0
0024 1E BD 00 03 | 0424 FE C6 E1 D0 | 0224 A0 58 A2 10 | 02A4 10 48 48 A9
0028 E6 E2 20 00 | 0428 39 A9 07 85 | 0228 A9 1F 20 67 | 02A8 57 8D 09 A0
002C F0 85 FC A2 | 042C E1 A9 81 8D | 022C 02 CA D0 F8 | 02AC A5 EA F0 23
0030 00 86 FF B5 | 0430 0E A0 A5 FE | 0230 20 F0 E9 A9 | 02B0 38 66 FE AD
0034 E9 95 E8 09 | 0434 6A 6A 29 1F | 0234 0D 20 72 02 | 02B4 07 A0 90 0B
0038 80 20 7B EF | 0438 F0 22 C9 1F | 0238 20 73 E9 C9 | 02B8 F0 15 A9 E9
003C A6 FF E8 E0 | 043C F0 1E C9 1B | 023C 0D F0 F1 C9 | 02BC 8D 04 A0 A9
0040 14 D0 EE A9 | 0440 D0 04 A9 20 | 0240 5B F0 15 AE | 02C0 00 F0 09 D0
0044 FB 8D 80 A4 | 0444 D0 18 05 FD | 0244 15 A4 E0 32 | 02C4 0A A9 86 8D
0048 A9 02 2C 82 | 0448 AA BD 68 00 | 0248 30 EB C9 20 | 02C8 04 A0 A9 01
004C A4 F0 0A 30 | 044C F0 19 E6 E3 | 024C F0 E2 C9 2D | 02CC 8D 05 A0 C6
0050 CE A9 7F 8D | 0450 A6 E3 9D 00 | 0250 D0 E3 20 72 | 02D0 EA 10 1A A9
0054 0E A0 4C 00 | 0454 03 E4 E2 D0 | 0254 02 4C 30 02 | 02D4 08 85 EA 86
0058 02 50 02 A9 | 0458 0E 20 91 E3 | 0258 A2 00 A9 0D | 02D8 EB A6 E9 E4
005C 80 29 80 8D | 045C A9 00 85 FD | 025C 20 72 02 BD | 02DC E8 F0 0C BD
0060 11 A4 4C 1F | 0460 10 05 A5 E4 | 0260 70 01 F0 CC | 02E0 00 03 C9 C0
0064 00 D3 FC FF | 0464 8D 09 A0 AD | 0264 E8 D0 F5 A4 | 02E4 F0 0C 85 FE
0068 00 45 00 41 | 0468 08 A0 68 A8 | 0268 E8 E6 E8 0A | 02E8 E8 86 E9 A6
006C 20 53 49 55 | 046C 68 AA 68 40 | 026C 09 C0 99 00 | 02EC EB 68 2C 04
0070 0D 44 52 4A | 0470 A9 07 85 E1 | 0270 03 60 A0 00 | 02F0 A0 40 A2 FF
0074 4E 46 43 4B | 0474 A9 7F 8D 0E | 0274 D9 68 00 F0 | 02F4 9A A9 00 8D
0078 54 5A 4C 57 | 0478 A0 58 4C 00 | 0278 03 C8 D0 F8 | 02F8 0B A0 4C 70
007C 48 59 50 51 | 047C 00 00 00 00 | 027C 98 48 29 20 | 02FC 04 00 00 00
0080 4F 42 47 00
0084 4D 58 56 00
0088 5D 33 00 2D
008C 20 27 38 37
0090 00 00 34 3B
0094 2C 5B 3A 28
0098 35 2B 29 32
009C 21 36 30 31
00A0 39 3F 5D 00
00A4 2E 2F 3D 00
```

<M>=10C 4C 70 04

Bild 1. Funkfernschreib-Empfangsprogramm mit Ausgabe auf den Thermodrucker des AIM-65. Startadresse ist 0470; dorthin zeigt auch der F1-Vektor (010C)

◄ Bild 2. Das Sendeprogramm wurde nahezu unverändert aus Heft 16/1979 der FUNKSCHAU übernommen. Geändert hat sich im wesentlichen nur der Bereich ab 02FA

Bild 3. ► Notwendige Änderungen im Sendeprogramm für eine Frequenzshift von 170 Hz statt 850 Hz

```
0000     *=2B3
02B3 AD LDA A006
02B6 90 BCC 02C3
02B8 10 BPL 02CF
02BA A9 LDA #58
02BC 8D STA A004
02BF A9 LDA #01
02C1 D0 BNE 02CC
02C3 F0 BEQ 02CF
```

schieht in der Interrupt-Routine, die die empfangenen Zeichen in den Bufferspeicher (FIFO, 0300...03FF) ablegt. Das Hauptprogramm – Startadresse ist 0470 – liest dieses ASCII-Zeichen aus dem FIFO wieder aus und stellt sich auf dem Display sowie – wenn das gewünscht wird – nun auch auf dem Thermodrucker dar.

An der Adresse 0001 läßt sich die Empfangsgeschwindigkeit einstellen. Für die drei üblichen Normgeschwindigkeiten ergeben sich folgende Werte:

45 Bd: hex 59
50 Bd: hex 51
75 Bd: hex 35

Während des Empfanges läßt sich der Drucker einschalten, indem man die Taste PRINT drückt, und wieder ausschalten, wenn man die Taste W kurz festhält und währenddessen PRINT antippt. Mit Taste „Escape" erfolgt ein Sprung an die Adresse 0200 – zu einem RTTY-Sendeprogramm, das weitgehend unverändert aus Heft 16/1979 der FUNKSCHAU übernommen wurde und in *Bild 2* ebenfalls als Hex-Dump wiedergegeben ist. Seine Baudrate läßt sich an der Adresse 02AB einstellen; dabei werden die gleichen Konstanten verwendet, wie sie gerade für das Empfangsprogramm angegeben wurden. Während des Sendens läßt sich der Drucker mit Control PRINT ein- und ausschalten. Taste F2 dient zum Rückschalten in den Empfangsmodus, sobald das letzte eingetippte Zeichen aus dem Sende-FIFO (ebenfalls 0300...03FF) ausgegeben ist. Die Taste F1 erlaubt im Sendemodus den Aufruf eines Standard-Textes (Standort, Rufzeichen usw.), der wie folgt vom Monitorprogramm aus eingegeben werden kann:

1. Editor mit Taste E initialisieren: FROM = 170, TO = 1E0.

2. Standardtext eingeben und zweimal (!!!) Return drücken.

3. Jetzt mit der Escape-Taste aus dem Editor aussteigen und das RTTY-Sende/Empfangsprogramm mit F1 starten.

Die vorliegende Programmversion erzeugt beim Senden am Port PB 7 die Mark-Frequenz 2125 Hz und die Space-Frequenz 1275 Hz. *Bild 3* zeigt die notwendigen Änderungen für 170 Hz Shift. Will man außerdem noch die Mark-Space-Lage vertauschen („Reverse"), so ist an der Adresse 02B6 einfach 90 statt B0 zu schreiben. ■

Hexadezimal-Dezimal-Umwandlung mit TI-59

Einige Male wurden schon Zahlenumwandlungs-Programme in der FUNKSCHAU vorgestellt. Oft haftet ihnen der Nachteil an, daß die Ausgabe hexadezimaler Zahlen nicht im gewohnten Format, sondern z.B. mit untereinanderstehenden Hex-Ziffern erfolgt. Das im *Bild* aufgelistete TI-59-Programm gestattet die Eingabe einer dezimalen Zahl und gibt sie auf dem Drucker PC-100 hexadezimal in gewohnter Schreibweise aus. Das Programm erklärt sich weitgehend selbst. Es belegt die Register 0, 1, 2, 3, 5, 11, 12 und 13; außerdem wird Flag 1 benutzt. Bei einer Kopplung mit einem anderen Programm muß Sorge getragen werden, daß die Speicherlöschung mit CMS (Adresse 206) durch andere Befehle ersetzt wird.

Werner Pötzschmann

```
000  43 RCL   040  02  02   080  69 OP   120  01  01   160  54  )   191  01  01
001  00  00   041  95  =    081  25  25   121  07  7    161  65  X   192  01  1
002  69 OP    042  59 INT   082  06  6    122  32 X:T   162  02  2   193  07  7
003  06  06   043  42 STO   083  32 X:T   123  61 GTO   163  54  )   194  04  4
004  68 NOP   044  01  01   084  43 RCL   124  00  00   164  42 STO   195  06  6
005  42 STO   045  65  X    085  05  05   125  63  63   165  05  05   196  02  2
006  01  01   046  01  1    086  77  GE   126  32 X:T   166  43 RCL   197  04  4
007  00  0    047  06  6    087  01  01   127  05  5    167  12  12   198  06  6
008  42 STO   048  45 YX    088  38  38   128  94 +/-   168  65  X   199  04  4
009  02  02   049  43 RCL   089  00  0    129  42 STO   169  01  1   200  00  0
010  01  1    050  02  02   090  48 EXC   130  13  13   170  00  0   201  00  0
011  06  6    051  95  =    091  13  13   131  22 INV   171  45 YX   202  69 OP
012  22 INV   052  52 EE    092  44 SUM   132  86 STF   172  43 RCL   203  02  02
013  49 PRD   053  22 INV   093  11  11   133  01  01   173  05  05   204  69 OP
014  01  01   054  52 EE    094  01  1    134  32 X:T   174  54  )   205  05  05
015  48 EXC   055  22 INV   095  00  0    135  61 GTO   175  52 EE   206  47 CMS
016  01  01   056  44 SUM   096  00  0    136  00  00   176  22 INV   207  98 ADV
017  59 INT   057  00  00   097  49 PRD   137  69  69   177  52 EE   208  92 RTN
018  48 EXC   058  01  1    098  11  11   138  00  0    178  69 OP   209  76 LBL
019  01  01   059  05  5    099  22 INV   139  48 EXC   179  04  04   210  11  A
020  69 OP    060  32 X:T   100  86 STF   140  13  13   180  02  2   211  42 STO
021  21  21   061  69 OP    101  01  01   141  44 SUM   181  03  3   212  00  00
022  22 INV   062  32  32   102  29 CP    142  12  12   182  01  1   213  01  1
023  97 DSZ   063  43 RCL   103  43 RCL   143  01  1    183  07  7   214  06  6
024  01  01   064  01  01   104  02  02   144  00  0    184  04  4   215  01  1
025  00  00   065  22 INV   105  77  GE   145  00  0    185  04  4   216  07  7
026  32  32   066  77  GE    106  00  00   146  49 PRD   186  01  1   217  04  4
027  69 OP    067  01  01   107  32  32   147  12  12   187  03  3   218  06  6
028  22  22   068  11  11   108  61 GTO   148  61 GTO   188  01  1   219  02  2
029  61 GTO   069  44 SUM   109  01  01   149  00  00   189  06  6   220  04  4
030  00  00   070  13  13   110  51  51   150  99  99   190  69 OP   221  69 OP
031  10  10   071  06  6    111  87 IFF   151  43 RCL               222  04  04
032  29 CP    072  44 SUM   112  01  01   152  11  11               223  61 GTO
033  43 RCL   073  13  13   113  01  01   153  69 OP                224  00  00
034  00  00   074  01  1    114  26  26   154  03  03               225  00  00
035  55  ÷    075  00  0    115  03  3    155  01  1                226  00  0
036  01  1    076  00  0    116  94 +/-   156  00  0                227  00  0
037  06  6    077  22 INV   117  42 STO   157  75  -
038  45 YX    078  49 PRD   118  13  13   158  43 RCL
039  43 RCL   079  13  13   119  86 STF   159  05  05

 1234567890.      DEZI
HEXADEZI= 499602D2
```

Komfort-Programm für die Zahlenumwandlung mit dem Rechner TI-59

Eine Morseschreibmaschine

Über den Sinn von Morsedecodern wollen wir uns hier nicht den Kopf zerbrechen; schließlich hat Telegrafie heute nur noch dort Sinn, wo es darum geht, aus einem hohen Störpegel noch ein schwaches Signal herauszuhören, und da kann der beste Morsedecoder nicht mehr mit. Sinnvoll ist es dagegen, Morsezeichen vom Terminal aus zu schreiben. Dazu dient hier der Mikrocomputer KIM-1 zusammen mit einem ASCII-Terminal.

Das hier beschriebene 6502-Maschinencode-Programm setzt jeden von einem ASCII-Terminal aus getippten Buchstaben in das entsprechende Morsezeichen um. Über einen I/O-Port des Mikrocomputers KIM-1 kann direkt z. B. ein Kurzwellensender angesteuert werden; wahlweise ist auch eine Ausgabe als Nf-Ton möglich. Die Eingabe kann sehr schnell erfolgen, da sich der Mikrocomputer einen 255 Zeichen umfassenden Zwischenspeicher aufbaut und diesen mit einer beliebig programmierbaren Geschwindigkeit als Telegrafie (CW) ausgibt.

Bild 1 zeigt das komplette Programm. Es verwendet die in Heft 7/1979, Seite 396, bereits näher besprochene Interrupt-Routine zum Empfang der ASCII-Zeichen, die hier nur innerhalb des „Hex-Dump" wiedergegeben ist. Bild 1 enthält auch die Tabelle zur Umwandlung der ASCII- in Morsezeichen (Adressen 006B...00AA). Es ist zu beachten, daß unbedingt die NMI-Leitung des KIM-1 mit dem Port PB 7 zu verbinden ist, da sonst kein Zeichen vom Terminal empfangen werden kann.

Das Kernstück der Morseschreibmaschine ist der in *Bild 2* gezeigte Programmteil. Er gibt ein im Akkumulator des Mikroprozessors 6502 stehendes ASCII-Zeichen in Telegrafie am Port PA 0 aus; dies geschieht als digitales Signal, wobei H-Pegel an PA 0 dem Zustand „Sender aus" und L-Pegel dem Zustand „Sender ein" entspricht.

Für Test- und Übungszwecke kann es sinnvoll sein, nicht ein digitales Signal zum Tasten des Senders zu erzeugen, sondern eine Tonfrequenz, die dann z. B. mit Hilfe der im KIM-1-Handbuch beschriebenen Schaltung (PNP-Transistor als Lautsprecher-Treiber) hörbar gemacht werden kann. Dazu sind die in *Bild 3* angegebenen Adressen zu ändern.

Die Eigenschaften der abgegebenen CW-Zeichen können durch Ändern bestimmter Zero-Page-Adressen den individuellen Erfordernissen angepaßt werden. Die Standard-Inhalte dieser Zellen sind ebenfalls in Bild 2 angegeben. Zu bedenken ist, daß eine Änderung der Tonfrequenz auch die CW-Geschwindigkeit beeinflußt: Höhere Zahlenwerte ergeben niedrigere Frequenzen und Geschwindigkeiten.

Für Übungszwecke wird manchmal der Zeichenabstand über die Norm von drei Punktlängen hinaus verlängert. Dies kann hier durch Einschreiben der gewünschten Punktlängen in die Zelle 00E9 geschehen.

Der von dem Programmteil in Bild 3 erzeugte Nf-Ton hört sich leider ein wenig rauh an. Dies hängt damit zusammen, daß er im Abstand von einigen Millisekunden jeweils für die Dauer einiger Mikrosekunden von der ASCII-Eingabe-Interruptroutine unterbrochen wird. Ein absolut sauberer Ton ist daher nur zu erzeugen, indem man wie in Bild 2 ein rein digitales Signal erzeugt und damit einen Tongenerator steuert, wenn auch die „Unsauberkeit" des Software-Tones nur gering ist.

Der Vorteil einer Morseschreibmaschine wird beim praktischen Betrieb schnell deutlich: Man kann so schnell geben wie bei RTTY, und die Zeichen werden mit konstant exaktem Punkt-Strich- und Punkt-Pausen-Verhältnis gesendet; und das ist wiederum Voraussetzung für das Funktionieren von Morsedecodern – siehe oben.

Herwig Feichtinger

```
0028 29 3F     AND #3F
002A AA        TAX
002B BD 6B 00  LDA 006B,X
002E 0A        ASL A
002F D0 09     BNE 003A
0031 A5 E9     LDA E9
0033 85 F4     STA F4
0035 20 54 00  JSR 0054
0038 F0 DD     BEQ 0017
003A A2 01     LDX #01
003C 90 02     BCC 0040
003E A2 03     LDX #03
0040 86 F4     STX F4
0042 20 4F 00  JSR 004F
0045 A2 01     LDX #01
0047 86 F4     STX F4
0049 20 54 00  JSR 0054
004C 18        CLC
004D 90 DF     BCC 002E
004F A2 01     LDX #01
0051 8E 01 17  STX 1701
0054 A6 EE     LDX EE
0056 A0 80     LDY #80
0058 8C 00 17  STY 1700
005B 88        DEY
005C D0 FD     BNE 005B
005E CA        DEX
005F D0 F5     BNE 0056
0061 C6 F4     DEC F4
0063 D0 EF     BNE 0054
0065 A2 00     LDX #00
0067 8E 01 17  STX 1701
006A 60        RTS

00E9 03 PAUSE
00ED 80 TONFREQUENZ
00EE 60 GESCHWINDIGKEIT
```

Bild 2. Disassemblierte Auflistung des Programmteils für die Ausgabe eines CW-Zeichens zum Tasten eines Senders. Das im Akku stehende ASCII-Zeichen wird in den Morsecode umgesetzt

```
0000 A9 02 85 E6 85 E7 85 EB 85 EC 8D 0F 17 A9 00 8D
0010 FB 17 A9 AB 8D FA 17 A5 E6 38 E5 EB F0 F9 C6 E6
0020 A0 00 B1 E6 C9 20 30 EF 29 3F AA BD 6B 00 0A D0
0030 09 A5 E9 85 F4 20 54 00 F0 DD A2 01 90 02 A2 03
0040 86 F4 20 4F 00 A2 01 86 F4 20 54 00 18 90 DF A2
0050 01 8E 01 17 A6 EE A4 ED 88 D0 FD EE 00 17 CA D0
0060 F5 C6 F4 D0 EF A2 00 8E 01 17 60 4C 60 88 A8 90
0070 40 28 D0 08 20 78 B0 48 E0 A0 F0 68 D8 50 10 C0
0080 30 18 70 98 B8 C8 AC 80 54 80 80 80 56 80 80 80
0090 80 80 80 80 80 80 CE 8C 56 94 FC 7C 3C 1C 0C
00A0 04 84 C4 E4 F4 56 56 80 80 80 32 48 98 48 A5 E8
00B0 D0 16 2C 40 17 10 09 A9 40 8D 0E 17 68 A8 68 40
00C0 A9 09 85 E8 A9 A0 D0 F1 C6 E8 D0 0E A5 EA F0 06
00D0 C6 EB A0 00 91 EB A9 8E D0 DF 18 2C 40 17 10 01
00E0 38 66 EA 18 90 F0
```

Bild 1. Mit diesem kleinen Programm wird aus dem Mikrocomputer KIM-1 eine „Morseschreibmaschine"

```
0056 A4 ED     LDY ED
0058 88        DEY
0059 D0 FD     BNE 0058
005B EE 00 17  INC 1700
```

Bild 3. Programmänderung zum Erzeugen eines Nf-Tones am Port PA 0

Das Besondere an dem im folgenden Beitrag beschriebenen Morsezeichen-Decoder ist, daß für das Einlesen der Zeichen eine Interrupt-Routine verwendet wird. Dieses Verfahren ermöglicht nämlich die gleichzeitige Ausgabe auf Display und Thermodrukker der Computer AIM-65 und PC-100.

Dr. Martin Costabel

AIM-65 decodiert Morsezeichen

Die prinzipielle Arbeitsweise von Morsedecodern wird in FUNKSCHAU 1980, Heft 9, näher beschrieben. Dort findet sich unter anderem auch ein CW-Empfangsprogramm für den AIM-65, das sich mit dem in Heft 16/1979 veröffentlichten CW-Sendeprogramm [1] zu einem kompletten Morseterminal kombinieren läßt. Die Arbeitsweise ist „straightforward" – gut durchschaubar also, führt aber zu dem Nachteil, daß aus Zeitgründen die Ausgabe nur auf dem Display, nicht aber mit dem Thermodrucker des AIM-65 möglich ist. Der Ausdruck einer Zeile dauert nämlich länger als die Pause zwischen zwei Zeichen [2, 3].

Ähnlich wie bei dem in diesem Heft beschriebenen RTTY-Programm mit Druckerausgabe wurde deshalb ein alternatives CW-Empfangsprogramm entwickelt, das dieses Problem durch eine vom 6522-Timer periodisch ausgelöste Interrupt-Empfangsroutine löst. Dabei wird der Adressenbereich 0500...05FF als Zwischenspeicher verwendet (FIFO = first in, first out). *Bild 1* zeigt zunächst das Hauptprogramm. Es wird mit der Taste F1 an der Adresse 0200 gestartet und initialisiert den Interrupt-Timer im VIA-Baustein 6522.

Bild 2 zeigt die Interrupt-Routine, die die fertigen ASCII-Zeichen in den FIFO abspeichert. Sie enthält auch eine automatische Geschwindigkeitsanpassung (041D...042D), die sich ausschalten läßt, indem man 041B von 90 auf D0 ändert. Als Codeumwandlungstabelle dient *Bild 3*; das empfangene Morsezeichen dient dabei als Tabellenindex, und die Datenbytes in der Tabelle stellen die jeweiligen ASCII-Zeichen dar.

Vor dem Start des Programms mit F1 ist ein CW-Tondecoder an den Port PA 0 anzuschließen. Dazu eignet sich zum Beispiel die in Heft 16/1979 der FUNKSCHAU veröffentlichte AIM-Erweiterungsplatine mit der PLL-Schaltung NE 567. Die Polarität ist hier für log. 0 = Signal, log. 1 = kein Signal ausgelegt und läßt sich für die Verwendung anderer Decoder an der Adresse 040B ändern, indem man F0 statt D0 einschreibt.

Nach dem Start erscheint im Display zunächst die Frage „TEMPO?", die mit einer vorgewählten Geschwindigkeit in Form einer stets zweistelligen Zahl zwischen 01 und 3F beantwortet werden muß. Es ist empfehlenswert, eher eine etwas höhere Geschwindigkeit (kleinere Zahl) zu wählen, da dann die automatische Anpassung schneller funkioniert. Bei gestörtem Empfang ist es ohnehin besser, ganz auf die Automatik zu verzichten.

In der Praxis konnten bereits Morsetelegramme kommerzieller Sender mit bis zu 600 Buchstaben pro Minute empfangen werden – das entspricht etwa zwei Druckzeilen pro Sekunde. Für den Test eignen sich recht gut die im Bereich um 8 MHz zu hörenden Wetterstationen.

```
010C 4C 00 02 00
0200 20 13 EA A9
0204 54 20 7A E9
0208 A9 45 20 7A
020C E9 A9 4D 20
0210 7A E9 A9 50
0214 20 7A E9 A9
0218 4F 20 7A E9
021C 20 3E E8 20
0220 D4 E7 20 73
0224 E9 20 84 EA
0228 20 73 E9 20
022C 84 EA 0A 85
0230 FE 4A 4A 85
0234 FD 20 13 EA
0238 A9 00 85 E5
023C 85 E6 85 E7
0240 85 E2 85 E3
0244 8D 03 A0 8D
0248 04 A0 8D 00
024C A4 A9 04 8D
0250 01 A4 A9 01
0254 85 E4 A9 40
0258 8D 0B A0 A9
025C C0 8D 0E A0
0260 A9 06 8D 05
0264 A0 EA A9 80
0268 A6 E2 E4 E3
026C F0 05 BD 00
0270 05 E6 E2 C9
0274 80 F0 19 20
0278 00 F0 85 FC
027C A2 00 86 FF
0280 B5 E9 95 E8
0284 09 80 20 7B
0288 EF A6 FF E8
028C E0 14 D0 EE
0290 A9 FB 8D 80
0294 A4 A9 02 2C
0298 82 A4 F0 0A
029C 30 C8 A9 7F
02A0 8D 0E A0 6C
02A4 FC FF 50 02
02A8 A9 80 29 80
02AC 8D 11 A4 4C
02B0 66 02 E0 14
```

```
0400 EA 48 8A 48
0404 98 48 A9 01
0408 2C 0F A0 D0
040C 04 E6 E7 D0
0410 65 E6 E6 A5
0414 E7 C5 FD 90
0418 1E C5 FE 90
041C 11 18 65 FD
0420 65 FD 65 FE
0424 4A 4A EA 85
0428 FE 4A 4A 85
042C FD 38 26 E4
0430 4C 70 04 EA
0434 EA EA EA 65
0438 E6 AA 85 E6
043C A9 00 E4 FE
0440 90 32 A6 E4
0444 BD 00 03 C9
0448 20 D0 0F C6
044C E5 F0 07 E6
0450 E5 E6 E5 4C
0454 6C 04 A2 02
0458 D0 02 A2 00
045C 86 E5 E6 E3
0460 A6 E3 9D 00
0464 05 E4 E2 D0
0468 03 20 91 E3
046C A9 01 85 E4
0470 A9 00 85 E6
0474 85 E7 AD 04
0478 A0 68 A8 68
047C AA 68 40 00
```

Bild 2. Die Interrupt-Routine empfängt die Morsezeichen, wandelt sie in ASCII um und schiebt sie in den Bufferspeicher

```
0300 00 20 45 54
0304 49 41 4E 4D
0308 53 55 52 57
030C 44 4B 47 4F
0310 48 56 46 55
0314 4C 41 50 4A
0318 42 58 43 59
031C 5A 51 4F 48
0320 35 34 00 33
0324 00 00 00 32
0328 00 00 23 00
032C 00 00 00 31
0330 36 3D 2F 00
0334 00 2A 00 00
0338 37 00 00 00
033C 38 00 39 30
0340 00 00 00 00
0344 00 00 00 00
0348 00 00 00 00
034C 3F 00 00 00
0350 00 00 22 00
0354 00 2E 00 00
0358 00 00 00 00
035C 00 00 27 00
0360 00 2D 00 00
0364 00 00 00 00
0368 00 00 00 00
036C 00 2F 00 00
0370 00 00 00 2C
0374 00 00 00 00
0378 3A 00 00 00
```

Bild 3. Diese Datentabelle dient der Umwandlung des Morsecodes in ASCII-Zeichen

◄ Bild 1. Mit der Taste F1 (Vektor: 010C) kann man das Programm an der Adresse 0200 starten. PRINT schaltet den Drucker ein, und PRINT bei gedrückter Taste W schaltet ihn wieder aus

Literatur

[1] *Feichtinger, H.:* AIM-65 als Morse-Schreibmaschine. FUNKSCHAU 1979, Heft 16.

[2] *Feichtinger, H.:* So funktionieren CW-Decoder. FUNKSCHAU 1980, Heft 9.

[3] *Feichtinger, H.:* AIM-65 empfängt Morsezeichen. FUNKSCHAU 1980, Heft 9.

Menue

Wie man Programme auch ohne Bandzählwerk wiederfindet

Das im *Bild* als Assemblerlisting wiedergegebene 6502-Maschinenprogramm für den AIM-65 alias PC-100 arbeitet als „Manager" für mehrere Programme, die nacheinander mit dem für eventuelle Änderungen erforderlichen Abstand auf eine gemeinsame Kassette gespeichert werden sollen. Da preiswerte Kassettenrecorder gewöhnlich nicht vollständig fernsteuerbar sind – nur der Motor läßt sich vom AIM-65 ein- und ausschalten –, gibt das Programm dem Benutzer Anweisungen, welche Tasten er auf dem Recorder zu drücken hat, um möglichst schnell die gewünschte Bandstelle zu finden.

Sinnvollerweise speichert man das Menue-Programm an den Anfang jeder Kassette, so daß es immer zuerst geladen wird. Wer über einen AIM-Assembler verfügt, kann dies direkt vom Assembler aus tun (OBJ-OUT=T), wobei auf die Kassette auch ein neuer Display-Link-Vektor gespeichert wird. Beim Laden springt das AIM-Monitorprogramm dann zum Menue-Programm, das dadurch vollautomatisch gestartet wird und zunächst den Display-Vektor wieder auf seinen Normalwert zurücksetzt. Außerdem wird der Motor des Recorders abgeschaltet.

Dann weist der Mikrocomputer den Benutzer an, die Taste „Schneller Vorlauf" am Recorder zu drücken und eine Hex-Ziffer (1...F) zur Kennzeichnung der gewünschten Bandstelle einzugeben. Sobald das geschehen ist, wird der Recorder-Motor für eine bestimmte Zeitdauer eingeschaltet, bis die gewünschte Bandstelle erreicht ist, und es erfolgt ein Rücksprung zum Monitorprogramm des AIM-65. Natürlich wäre es auch möglich, direkt zur Laderoutine weiter zu springen; dies funktioniert aber nur, solange man keine Editor- oder Basic-Files laden will. Deshalb wurde hier der Weg vorgezogen, daß der Benutzer selbst den Ladevorgang einleitet, z.B. indem er L drückt. Außerdem kann das Menue-Programm dadurch nicht nur zum Laden, sondern auch zum Abspeichern verwendet werden.

Leser, die kein Assembler-ROM in ihrem AIM-65 stecken haben, müssen leider auf die Eigenschaft des „Autostart" verzichten, denn der Display-Link-Vektor läßt sich nicht von Hand ändern – das System würde sich sofort „aufhängen". Eine Alternative wird in diesem Heft beschrieben. Eine gewisse Erleichterung stellt es auch dar, wenn man den Vektor für die Taste F1 auf 0F00 richtet (010C: 4C 00 0F) und mit auf Band aufzeichnet. Nach dem Laden des Menue-Programms braucht man dann nur noch F1 zu drücken.

Man beachte auch bei der Eingabe des Programms ohne Assembler, daß die Hex-Codes der Texte nicht vollständig im Bild aufgelistet sind. Die Codes für die einzelnen Buchstaben ab Adresse 0F54 lauten: 50 52 45 53 53 20 46 46 57 44 3B 50 47 4D 20 4E 52 2E 3F 00 20 50 52 45 53 53 20 50 4C 41 59 20 41 4E 44 20 4C 4F 41 44 00.

Durch Ändern der Daten an der Adresse 0F2C ist es möglich, die Vorlaufdauer bzw. den Abstand zwischen den einzelnen Programmen auf einer Kassette zu ändern und damit der durchschnittlichen Aufzeichnungsdauer anzupassen. *Herwig Feichtinger*

```
A408          PASS 1
A408          PASS 2
0000
0000          ;◆◆◆◆◆ MENUE ◆◆◆◆◆
0000 MONI             =$E182       ;REENTER
0000 DRB              =$A800
0000 CNTH             =$FD
0000 CNTL             =$FE
0000 NR               =$FF
0000 CRLF             =$E9F0
0000 LL               =$E8FE       ;KB=INP
0000 HEX              =$EA7D       ;ASCII-HEX
0000 IN               =$E993       ;INP KEY
0000 OUT              =$E9BC       ;PRINT
0000                  *=$F00
0F00 MENUE A905       LDA #5       ;RESET
0F02       8D06A4     STA DILNK    ;DISPL LINK
0F05       A9EF       LDA #$EF
0F07       8D07A4     STA DILNK+1
0F0A       20FEE8     JSR LL
0F0D       20F0E9     JSR CRLF
0F10       A904       LDA #4
0F12       8D00A8     STA DRB      ;TAPES OFF
0F15       A200       LDX #0
0F17       20480F     JSR PRT      ;FAST FORWD
0F1A       2093E9     JSR IN
0F1D       207DEA     JSR HEX      ;FFWD TIME
0F20       85FF       STA NR
0F22       A934       LDA #$34
0F24       8D00A8     STA DRB      ;TAPES ON
0F27 TMG   C6FF       DEC NR
0F29       F011       BEQ RDY
0F2B       A90E       LDA #14      ;DELAY 5SEC
0F2D       85FD       STA CNTH
0F2F DEL   88         DEY
0F30       D0FD       BNE DEL
0F32       C6FE       DEC CNTL
0F34       D0F9       BNE DEL
0F36       C6FD       DEC CNTH
0F38       D0F5       BNE DEL
0F3A       F0EB       BEQ TMG      ;UNCOND.
0F3C RDY   A904       LDA #4       ;TAPE OFF
0F3E       8D00A8     STA DRB
0F41       E8         INX
0F42       20480F     JSR PRT
0F45       4C82E1     JMP MONI
0F48 PRT   BD540F     LDA TXT,X    ;OUTP TEXT
0F4B       F006       BEQ RTN      ;UNTIL 00
0F4D       20BCE9     JSR OUT
0F50       E8         INX
0F51       D0F5       BNE PRT
0F53 RTN   60         RTS
0F54 TXT   5052       .BYT 'PRESS FFWD'
0F5E       3B50       .BYT ';PGM NR.?'
0F67       00         .BYT 0,' PRESS PLAY'
0F68       2050
0F73       2041       .BYT ' AND LOAD',0
0F7C       00
0F7D                  *=$A406
A406 DILNK 000F       .WOR MENUE   ;AUTOSTART
A408                  .END
A408          ERRORS= 0000
```

Das Programm kann man zeitsparend hexadezimal mit der Taste „/" ab Adresse 0F00 eingeben (A9 05 8D 06...) – mit einer Ausnahme: Der DILNK-Vektor läßt sich nicht von Hand ändern!

GWK GESELLSCHAFT
FÜR TECHNISCHE ELEKTRONIK mbH.

DIE SPEZIALISTEN FÜR 6502 UND 6809
in Hardware, Software und Systementwicklung

MIKRO-COMPUTER
Rockwell AIM 65
SIEMENS PC 100
SIEMENS PC 1000

EURO-BOARD-EXPANSION-SYSTEME
CPU, Speicher RAM/ROM, EPROM-Programmer, Input/Output, AD/DA-Converter, Power Supplys, Floppy-Disk-Controller, Video-Interface

SOFT-WARE
Standard- und Anwendersoftware.
Basic, Basic-Expansion, Texteditor,
Assembler, Development Aids, Disk-Operating-System

PERIPHERIE-GERÄTE
Floppy-Disk, Printer, Terminal, Monitor, Gehäuse usw.

DIENSTLEISTUNG
Beratung und Problemanalyse.
Erstellung kundenspezifischer Systeme.
Durchführung von Entwicklungsaufträgen.

Bitte fordern Sie ausführliches Informationsmaterial an.

GWK GESELLSCHAFT
FÜR TECHNISCHE ELEKTRONIK mbH.

Asternstr. 2, 5120 Herzogenrath, Telefon 0 24 06/6 23 94

HEATH ZENITH **data systems**

personal-computer wh-89 + **zeilendrucker wh-14**

Beide Komponenten ergänzen sich zu einem vielseitigen Datenverarbeitungs-System.

Der Computer:
- Zwei Z80 Mikroprozessoren
- Grundausbau 16K Byte, bis 48K erweiterungsfähig
- Mono-Floppy-Disk-Laufwerk
- 12-Zoll-Bildschirm mit 25 Zeilen je 80 Zeichen
- Professionelles Tastenfeld

Grundmodell
mit 16K RAM und Floppy-Disk
Bausatz: DM 5.254,–
Gerät: DM 6.650.–

Der Zeilendrucker:
- Mikroprozessorgesteuert (3870)
- Max. Geschwindigkeit 135 Zeichen/Sek.
- Zeichensatz Groß-/Kleinschreibung mit 96 Zeichen ASCII
- Zeilenlänge 80, 96 und 132 Zeichen
- Zeilenabstand 6 Zeilen/Zoll oder 8 Zeilen/Zoll (anwählbar)
- Zeichenmatrix 5 x 7 Punkt
- Schnittstelle EIA RS-232 C mit einer Baudrate von 110 – 4800

Bausatz: DM 1.570.–
Gerät: DM 2.595.–
(Alle Preise verstehen sich incl. Mehrwertsteuer)

HEATHKIT Coupon C

HEATHKIT Bausatz-Elektronik GmbH
Ausstellungs- und Service-Zentrum
Robert-Bosch-Straße 32 – 38
6072 Dreieich-Sprendlingen
Postfach 102060
Telefon 06103/3808

Senden Sie mir bitte kostenlos ausführliches Informationsmaterial.
Name
Straße
PLZ Ort

Erweitern Sie Ihren PET!

Speicherkarte 32 K
für PET 2001, Erweiterung um 32 K im PET-Gehäuse ohne zusätzliche Spannungsversorgung, mit Steckadapter **1.495,–**

große Zusatztastatur
2 Return-, 3 Shift- und 2 Space-Tasten
ohne Gehäuse **349,–**
mit Gehäuse **398,–**

Elbatex GmbH
Handelsunternehmen für Elektronik
Cäcilienstraße 24 · D-7100 Heilbronn · Tel. 07131/8 90 01 · Telex 7 28362

nascom
DER MICROCOMPUTER „TREND-SETTER"

neu

NASCOM-1

Ideales kostengünstiges System für Anfänger – mit folgenden Eigenschaften:
- Tastatur mit 48 Tasten in „QWERTY"-Anordnung, auf 58 Tasten erweiterbar.
- Z80 CPU, läuft bei 2 MHZ
- 2K Statische RAM
- 2K NAS-SYS MONITOR (PI/o chip nicht inklusiv)
- 16×48 Video-Interface für Standard-T.V. mit UFH-Ausgang
- Monitor-Interface
- Cassetten- und TTY-Interface (RS 232, 20 mA Loop) 2×8 Bit I/O Ports (PIO) TTL-Kompatibel für Drucker, Relais, Eingänge, Steuerung usw.
- Erweiterbar auf 64K über Buffer Board
- 3 Amp. Netzteil Bausatz, speziell für NASCOM-1 DM 158,20 inkl. MWSt.
- Buffer Bus Board: DM 158,20 inkl. MWSt.
- 8K Speicher KIT: DM 441,–– inkl. MWSt.
- 16K Speicher KIT: DM 655,40 inkl. MWSt.
- 32K Speicher KIT: DM 847,50 inkl. MWSt.

KIT DM 659,–
System DM 759,–
inkl. MWSt.

Die Preis-Sensation

Alle Erweiterungen des NASCOM-1 und des NASCOM-2 können untereinander ausgetauscht werden!

N.A.S.
Tel. 089/5233153/4 · Telex 0522061
Briennerstr. 56
D-8000 MÜNCHEN 2
Elektronische Halbleiter GmbH

AUFRÜSTEN!

Mit dem VERO-Speichererweiterungs-Paket
für alle COMMODORE-Computer mit neuem Betriebssystem

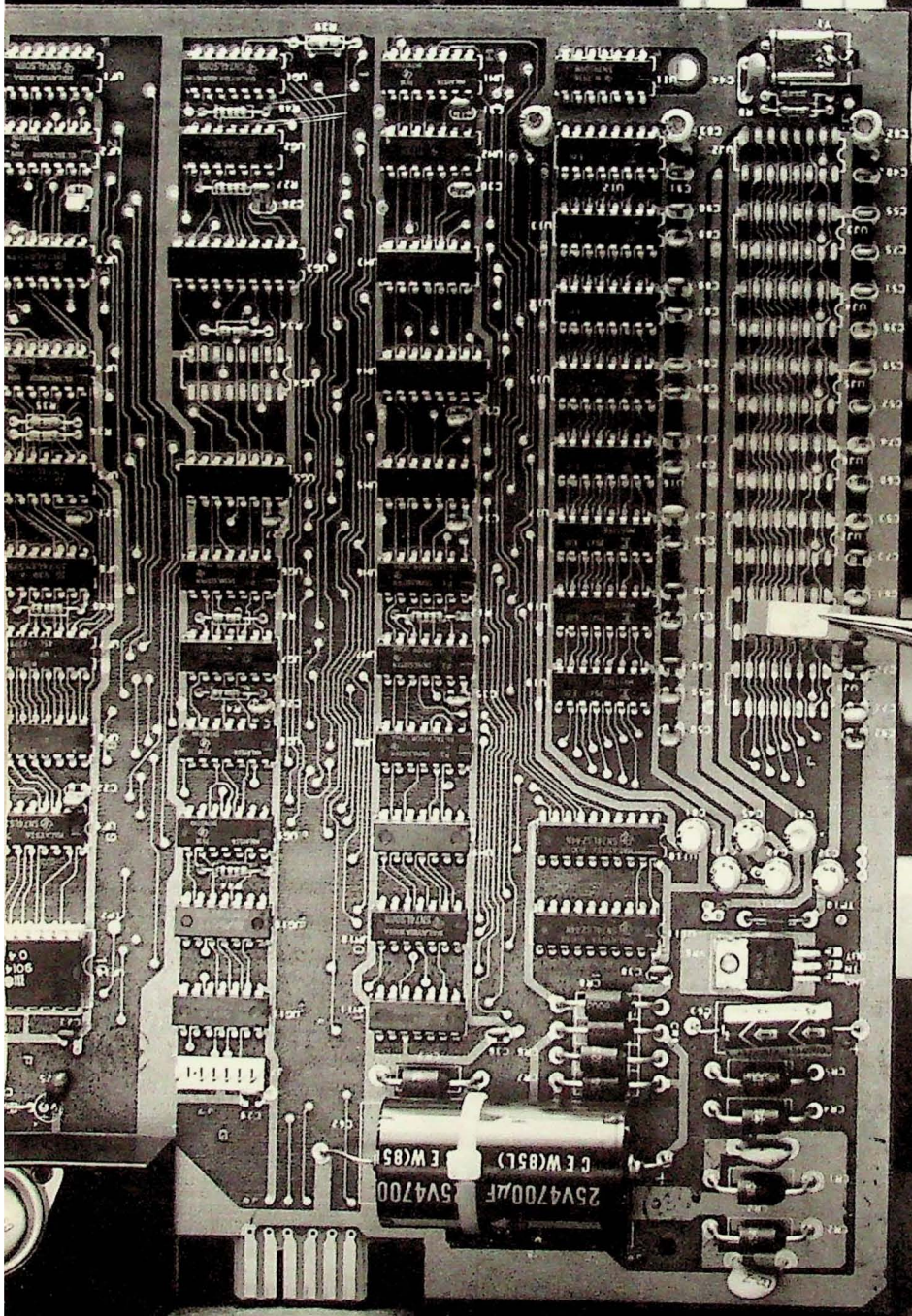
Macht aus Ihrem CBM 8 K einen vollwertigen CBM 16 K und 32 K. Oder aus 16 K einen vollwertigen 32 K - Ram.

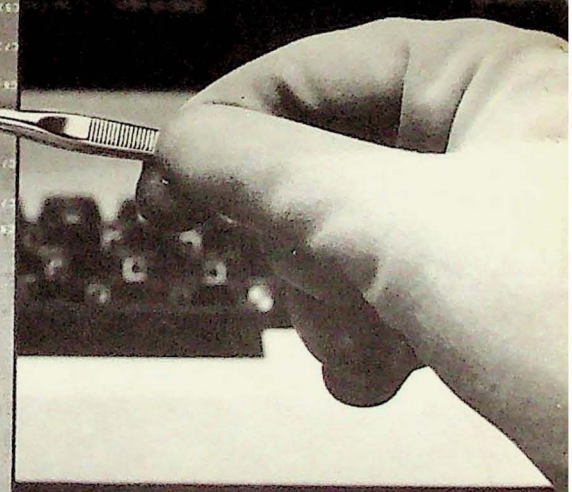
Der Leistungs- und Lieferungsumfang:
1) Aufrüstung der Original-CBM-Platine in unserem Meisterbetrieb

2) Das Vero-Diagnosepaket ist im Lieferumfang enthalten

3) 6 Monate Vollgarantie

Wieder eine echte VERO-Leistung!





Aufrüstung	auf 16 K	auf 32 K
von CBM 8 K	348,-	598,-
von CBM 16 K		498,-

Unglaublich preiswert!

Beim Neukauf empfiehlt sich die Kombination CBM 8 K (1.795,-) + Aufrüstung auf 32 K (598,-) = **2.393,- DM** incl. MwSt.

VERO COMPUTER
Qualifizierter Fachhandel für Computertechnik

VERO GmbH - Pontstraße 60/78 - Postfach 1778
5100 Aachen - Telefon 0241/2 33 45 und 3 38 06
Telex 08 32 389 vero d

Weitere Preise

Floppy 3040	2.850,-
Drucker 3022	1.948,-
Drucker 3023	1.695,-